REVISTA PARA USUARIOS DE

Drean commodore

AÑO 1 Nº 8 ₳ 2.30

REP. ARGENTINA



SUMARIO



NOTAS TECNICAS

PROGRAMAS

REVISION DE SOFT

SECCIONES FIJAS

En el número anterior comenzamos a describir información inédita sobre la Drean Commodore 16. Ahora veremos los registros del circuito integrado más importante de la máquina.

Continuamos describiendo el procedimiento para trabajar con archivos en los equipos Drean Commodore 16 y 64. En este número explicaremos el acceso relativo.



Utilizando el Print Shop estamos seguros que le enviaremos una tarjeta a la casa de Henry para su cumpleaños. Eso si no se atraviesa un Comando proveniente de Estelar 7.

Foto de tapa: gentileza de Skier's Magazine

Drean Commodore

AÑO 1 Nº 8 JULIO DE 1986

Director General
Ernesto del Castillo

Director Editorial
Cristian Pusso

Director Periodistico
Fernando Flores

Director Financiero:
Javier Campos Malbrán

Arte y Diagramación
Fernando Amengual
Tamara Migelson

Coordinador
Ariel Testori

Redacción
Cristian Parodi

Fotografia
Victor Grubicy

Departamento de Avisos
Oscar Devoto

Departamento de Publicidad
Guillermo Gonzalez Aldalur

Drean Commodore es una Revista mensual editada por editorial PROEDI S.A., Parana 720, 5º Pis. (1017) Buenos Aires. Tel.: 46-2886 y 49-7130. Reg. Nac. de la Prop. Intelectual E.T., M. Registrada.

Precio de este ejemplar. ₳ 2.30
Impresión: Calcotam. Fotocromo tapa: Columbia. Fotocomposicion: Van Waveren.
Los ejemplares atrasados se venderán al precio del último numero en circulación.

Distribuidor en Capital: Martino, Juan de Garay 358. P.B. Capital. Distribuidor interior: DGP. Hipólito Yrigoyen 1450. Capital Federal, T.E. 38-9266/9800.

TED: ALGO MAS QUE UN CHIP

En el número anterior esta sección estuvo involuntariamente ausente. Seguimos explicando terminología y los contenidos de las notas técnicas.



Los usuarios de la Drean Commodore 16 comprobarán que continuamos brindando datos totalmente inéditos referentes a su equipo.

En este número comenzamos a describir los registros que constituyen el chip TED.

Un registro forma parte de la unidad central de procesamiento (CPU), "cerebro" de toda computadora.

Los registros colaboran con el funcionamiento de la CPU, recordando instrucciones, direcciones de memoria, datos, etc. Igualmente cada "gran chip" (como lo es el TED del C-16) dispone de registros que se encargan de realizar determinadas tareas.

Por ejemplo, alguno de ellos administran la memoria, controlan la pantalla, verifican los contadores, se encargan de la generación del sonido, etc.

Continuamos explicando cuál es el método para definir las teclas de función. Hablamos sobre la rutina IRQ que, 60 veces por segundo, se encarga de ver si se ha oprimido alguna tecla. Interrupt ReQuest (requerimiento de interrupción) es una de las rutinas más importantes de todos los equipos Drean Commodore.

En el número anterior ya habíamos mostrado cómo insertar una rutina para que se ejecutase periódicamente cada 1/60 segundos. De todas maneras, y aunque les parezca que no tiene nada que ver, les pedimos que lean la nota sobre el chip TED. Allí explicamos quien se encarga de generar la IRQ.

En esa nota hablamos sobre los contadores y cómo le indican al sistema operativo que se debe provocar una interrupción sobre la tarea que se está realizando, para ir a ver si se oprimieron teclas o si oprimió la tecla STOP.

Para los usuarios de la C-128 tenemos una nota en donde comentamos como redefinir el juego de caracteres de su computadora para que puedan, por ejemplo, definir símbolos que no se encuentran en ninguna otra.

Antes habíamos mencionado al sistema operativo. No lo descuidamos y seguimos describiendo como usar las rutinas del Kernal desde el Basic o desde el Assembler. Recuerden que este último (Assembler) es un lenguaje que nos permite desarrollar programas a nivel del código de máquina, lo único que entiende la computadora.

Y como éstas son subrutinas, luego de ser ejecutadas retornarán al programa principal, más precisamente a la instrucción siguiente al llamado.

En otra parte de la revista hablamos sobre manejo de archivos.

Esto nos permite almacenar datos en disco o en cassette.

A diferencia de la memoria, en donde todos los datos se pierden cuando le desconectamos la alimentación, los archivos permiten guardar información independientemente de la energía eléctrica.

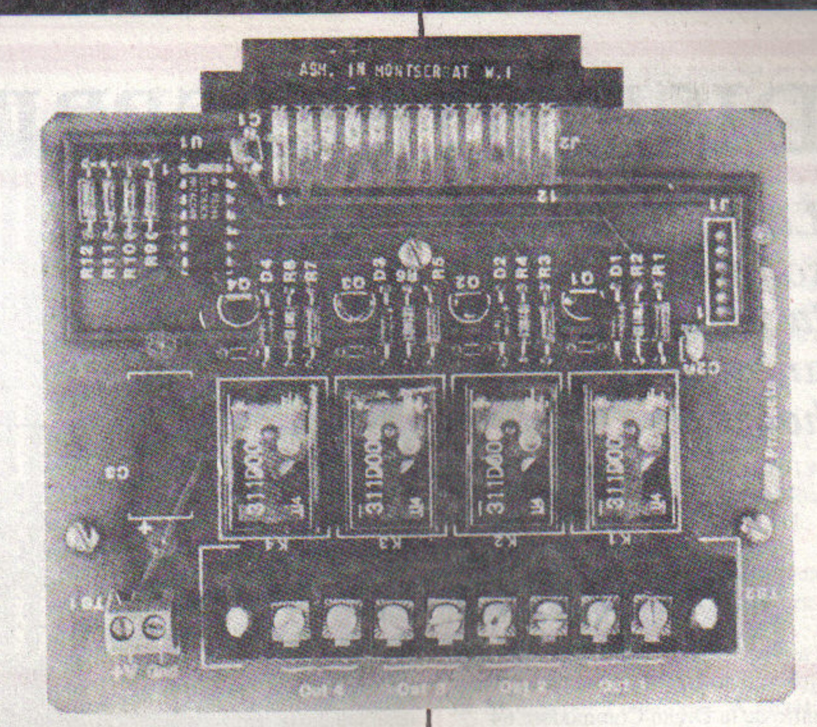
En la sección programas les presentamos tres listados totalmente inéditos. Uno de ellos les permitirá jugar al ajedrez.

Los otros dos son utilitarios para poder imprimir caracteres ampliados y en cualquier parte de la pantalla. Con el otro podrán almacenar en "slots" pantallas creadas por ustedes.

Un slot podemos definirlo, para esta aplicación, como un área de memoria determinada.

El último cartridge para 64

Así lo ha titulado la crítica del Reino Unido. Premiado como el mejor utilitario de 1985 este cartridge permite las siguientes funciones:
1) Aumenta la velocidad del disco 6 veces a través de un Disk Turbo.
2) Aumenta la velocidad del Datassette 10 veces a través de un Tape Turbo.
3) Dispone de una interface Centronics la cual permite conectar cualquier tipo de impresora paralelo a la C-64.
4) Permite la creación de dibujos en alta resolución. Compatible con el Print Shop y Koala.
5) Aumenta la capacidad de la Drean Commodore 64 a 24 kb. Para eso suministra dos nuevos comandos (Memory write y memory read).
6) Aumenta los comandos disponibles del basic de la Drean Commodore 64 (DLORD, DSAVE, CATALOG, etc.).



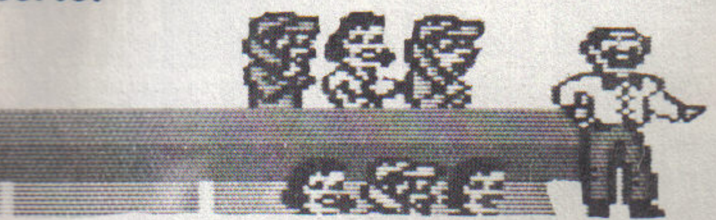
7) Suministra herramientas para la ejecución de edición de programas (Delete, Old, Renumber, Auto, etc.).
8) Permite programar las teclas de función.
9) Monitor Assembler residente.
10) Permite suspender la ejecución del proceso actual hasta que se oprima una tecla.
Por ahora este cartridge no se encuentra en nuestro país.
Esperemos que algún fabricante nacional pueda desarrollarlo.

Herramientas de trabajo

La popular empresa de software "Epyx" ha desarrollado para la Drean Commodore 64 un nuevo utilitario que suministra más de cien nuevos comandos.
Gracias a ellos se pueden escribir programas en Assembler, desarrollar gráficos comerciales, diseñar juegos, etc.
Además dispone de comandos orientados al manejo de gráficos en alta resolución.
Otro de los nuevos comandos es el "Joyst" a través del cual podemos tener absoluto control sobre un programa o proceso usando para ello el Joystick.
Para la parte de sonido se facilita el comando "Sound".
Con él podemos diseñar melodías completas y, si lo deseamos, es posible crear el pentagrama con las notas usadas.
Por ahora no se encuentra disponible en nuestro mercado.

DISEÑO DE SPRITES

Las posibilidades gráficas de la 64 son muchas. Entre ellas se encuentra la ventaja de poder crear gráficos animados. Aquí les explicamos cómo hacerlo.



Dentro de la Drean Commodore 64 existe un circuito integrado denominado VIC-II, el cual se encarga de realizar todas las tareas correspondientes a impresión de caracteres, gráficos, scrolling y el manejo de sprites.

Debido al Basic de nuestra computadora, la Versión 2.0 de Microsoft realizado en el '77 y de la que nunca se realizó una mejora, todo lo que respecta al manejo de gráficos (ya sean gráficos o sprites) es una tarea que debe realizarse a "mano".
Esto significa que debemos ingresar en direcciones establecidas y poner valores determinados.
El VIC está formado por varios registros, los cuales están representados por direcciones de memoria.
Nosotros sólo explicaremos las que involucran el trabajo con los sprites.
A no asustarse... allá vamos!!!

Definición de los Sprites

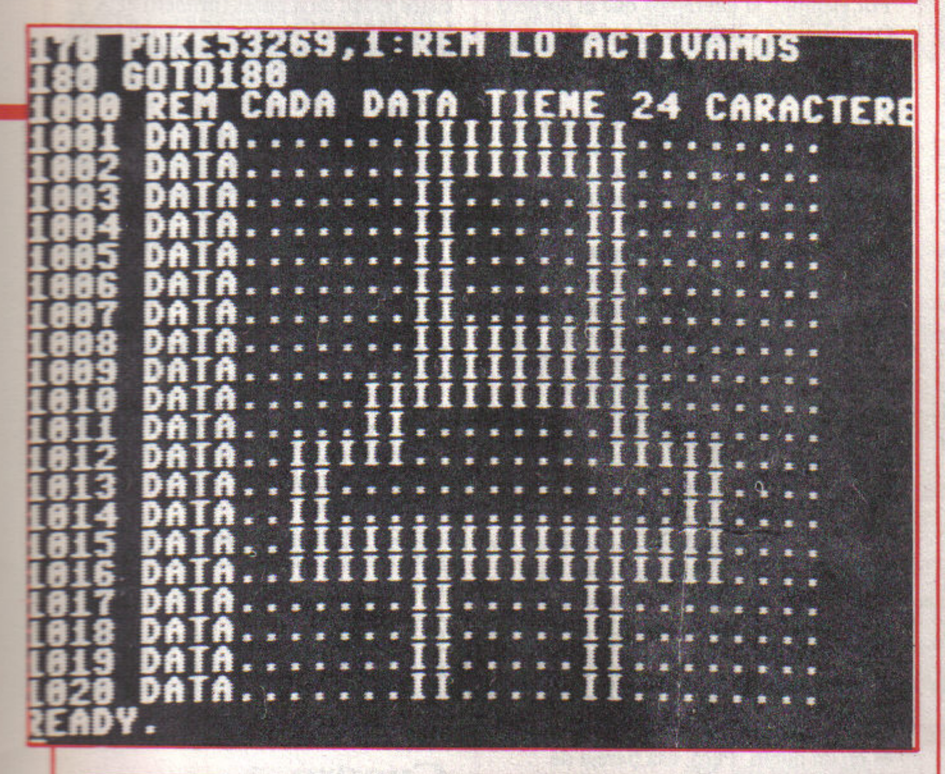
El VIC puede manejar 8 sprites en forma independiente. Ellos están numerados desde el 0 hasta el 7.
Cada uno debe ser definido con su correspondiente gráfico. Para ello se debe elegir el área de memoria donde se encontrarán los 63 bytes de datos que los definen.
Para ello se disponen de una serie de punteros. El VIC tiene 8 punteros (uno para cada sprite) que le indican la dirección inicial del área antes mencionada.
Estos se encuentran a continuación del área reservada para pantalla. La tabla 1 los describe.

Figura 1

Byte 1	Byte 2	Byte 3
Byte 4	Byte 5	Byte 6
. . . . .	. . . . .	. . . . .
. . . . .	. . . . .	. . . . .
Byte 58	Byte 59	Byte 60
Byte 61	Byte 62	Byte 63

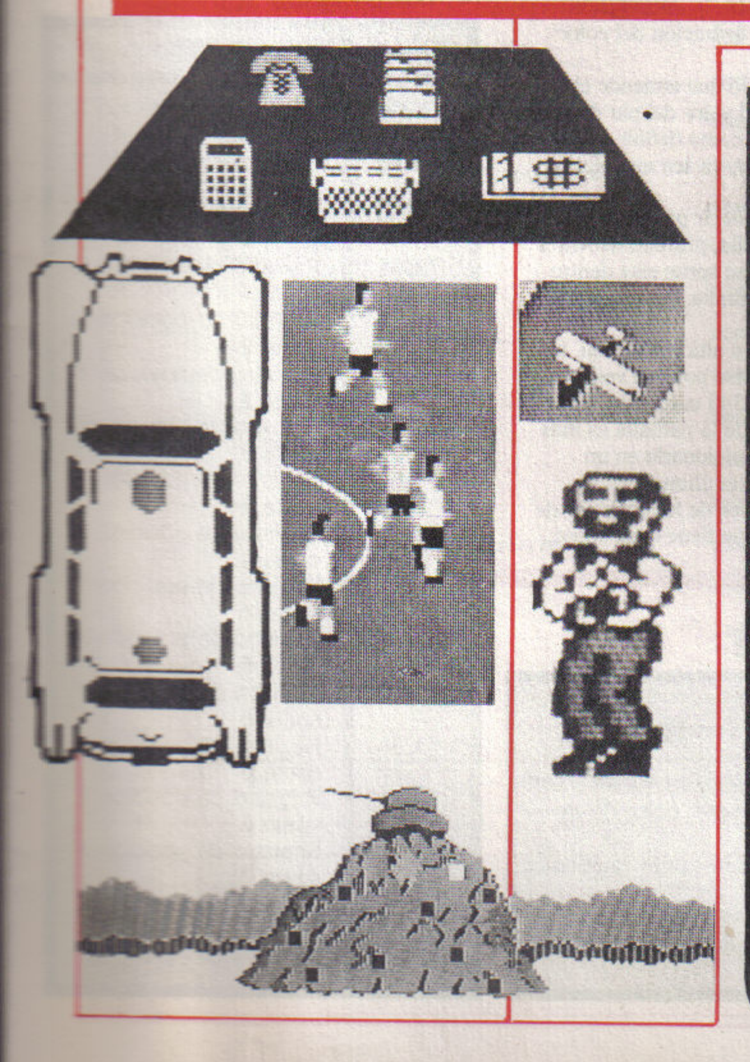
Figura 2

Valores:	128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1
bit:	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1																								
2																								
3																								
4																								
5																								
6																								
7																								
8																								
9																								
10																								
11																								
12																								
13																								
14																								
15																								
16																								
17																								
18																								
19																								
20																								
21																								



```
170 POKE53269,1:REM LO ACTIVAMOS
180 GOTO180
1000 REM CADA DATA TIEME 24 CARACTERE
1001 DATA.......IIIIIIIII........
1002 DATA.......IIIIIIIII........
1003 DATA.......II.....II........
1004 DATA.......II.....II........
1005 DATA.......II.....II........
1006 DATA.......II.....II........
1007 DATA.......II.....II........
1008 DATA.......IIIIIIIII........
1009 DATA.......IIIIIIIII........
1010 DATA.....IIIIIIIIIIII.......
1011 DATA.....II........II.......
1012 DATA..IIIII........IIIII....
1013 DATA..II..............II....
1014 DATA..II..............II....
1015 DATA..IIIIIIIIIIIIIIIIII....
1016 DATA..IIIIIIIIIIIIIIIIII....
1017 DATA.......II.....II........
1018 DATA.......II.....II........
1019 DATA.......II.....II........
1020 DATA.......II.....II........
READY.
```

Noten que el VIC sólo necesita un byte para determinar la dirección de inicio de los valores que definen a un sprite. Lo que él hace es tomar el valor almacenado en el puntero y multiplicarlo por 64. Por ejemplo, si en la dirección 2040 (puntero inicio definición sprite 0) se encuentra el valor 14, el VIC irá a buscar los datos a la dirección dada por 14*64=896. Esta dirección (896) corresponde al buffer del cassette, lo que implica que podemos almacenar valores sin interferir con el funcionamiento del intérprete (el rango de este buffer va desde la dirección 819 hasta la 1019).
A continuación debemos ingresar los valores que establecen la forma del sprite 0.
Como antes mencionamos se requieren 63 bytes. La figura 1 indica como se corresponden.
Para entender como se definen vean la figura 2, la cual representa una matriz de 21 filas por 3 columnas. A través de ella se puede representar un sprite.
La figura se diseña teniendo en cuenta que un "1" prende un punto y que un "0" no lo prende.
Una vez que completamos esa matriz, debemos codificar esos unos y ceros a



valores que sí entienda el VIC.
Para ello, simplemente, codificamos esos valores a su correspondiente en decimal.
Así se obtiene, por cada fila, tres valores decimales. Observen nuevamente la figura 2, y noten los valores que van desde 128 hasta 0 disminuyendo, en cada paso, a la mitad.
Por ejemplo, si la primer fila de un determinado gráfico es:
00000000 00000011 10000001
ellos se codifican a 0,3 y 129. Lo que hicimos fue ir sumando los valores de cada columna. Verifiquen ésto trasladando este valor a nuestra matriz.
Una vez definido el sprite, debemos indicarle al VIC el color que tendrá (si será multicolor o no), si estará ampliado, las coordenadas donde aparecerá y, finalmente, activarlo.
Para indicarle el color, debemos acceder a los registros de color de cada sprite. La figura 3 los representa. Algo que omitimos mencionar es que, el acceso a los registros, se hace a través de las sentencias PEEK y POKE.

Por ejemplo, si queremos que el sprite 0 (ya definido) tenga color negro, debemos hacer:

POKE 53287,0

Observen el manual del usuario para saber los códigos de los colores.
En caso de optar por formato multicolor, cada sprite puede tener hasta cuatro colores. Como consecuencia de ésto, la resolución horizontal disminuye a la mitad.
Para activar un sprite multicolor, se debe acceder al registro ubicado en la dirección 53276 ($D01C). Cada bit corresponde a un sprite. Así, si ponemos el valor 1, activaremos el sprite 0 a multicolor, con 2 activaremos el 1, con 4 el 3, etc.

Para desactivar este modo se pone un "0" en la posición deseada.
Como dijimos, trabajar en multicolor significa utilizar sprites definidos por una matriz de 12 por 21 puntos, en lugar de 24 por 21 puntos (observen que la resolución horizontal baja, efectivamente, a la mitad).
El motivo de ello es que el VIC, en este modo, supone que cada par de puntos es en un punto. Pero, esa pareja de puntos, le suministra la información necesaria para la definición del color del sprite.
La tabla 2 indica lo que entiende el VIC de acuerdo al valor del par de bits.
Hasta aquí vimos como definir un sprite y como darle color, ya sea un color o varios colores.
Para activarlo, es decir para que aparezca en pantalla, primero debemos asegurarnos que ese sprite esté dentro del rango de la pantalla, de otra forma no se verá.
Como se trabaja en alta resolución, un sprite puede moverse por una ventana de 249 líneas por 343 columnas.
Como la ventana de la pantalla es más chica, debemos posicionarlo en un sector dentro de esta última.
Eso se logra a través de los registros de posicionamiento que tiene cada sprite.

Estos están de a pares (primero el X y luego el Y). El registro X e Y del sprite 0 están en las direcciones 53248 y 53249; el del sprite 1 en las direcciones 53250 y 53151, etc. La tabla 3 representa estos registros.
Hay un último registro, en la dirección 53264 a continuación del registro Y del sprite 7, que se utiliza para poder posicionar un sprite en 343 puntos distintos.
Como cada registro X puede tener hasta 256 valores distintos, se utiliza este registro el cual suministra un octavo bit.
Cada bit de este registro es un octavo bit de un sprite. Por ejemplo, el bit 0 es el octavo bit del registro X del sprite 0; el bit 1 es el octavo bit del registro X del sprite 1, etc.
Es decir que si queremos trasladar un sprite más allá de la columna 255, debemos setear el bit correspondiente a este registro en 1.
Por ejemplo, para posicionar el sprite 0 en la columna 290 fila 70 debemos hacer:

Figura 3

Dirección	Descripción
53287 ($D027)	Registro de color sprite 0
53288 ($D028)	Registro de color sprite 1
53289 ($D029)	Registro de color sprite 2
53290 ($D02A)	Registro de color sprite 3
53291 ($D02B)	Registro de color sprite 4
53292 ($D02C)	Registro de color sprite 5
53293 ($D02D)	Registro de color sprite 6
53294 ($D02E)	Registro de color sprite 7

Cuadro 1

Dirección	Descripción
2040 ($07F8)	puntero inicio sprite 0
2041 ($07F9)	puntero inicio sprite 1
2042 ($07FA)	puntero inicio sprite 2
2043 ($07FB)	puntero inicio sprite 3
2044 ($07FC)	puntero inicio sprite 4
2045 ($07FD)	puntero inicio sprite 5
2046 ($07FE)	puntero inicio sprite 6
2047 ($07FF)	puntero inicio sprite 7

Cuadro 2

Par de Bits	Descripción
00	Transparente, color de la pantalla
01	Registro de sprite multicolor N° 0
10	Registro de color del sprite
11	Registro de sprite multicolor N° 1

Cuadro 3

Dirección	Descripción
53248	Registro de posición X sprite 0
53249	Registro de posición Y sprite 0
53250	Registro de posición X sprite 1
53251	Registro de posición Y sprite 1
53252	Registro de posición X sprite 2
53253	Registro de posición Y sprite 2
53254	Registro de posición X sprite 3
53255	Registro de posición Y sprite 3
53256	Registro de posición X sprite 4
53257	Registro de posición Y sprite 4
53258	Registro de posición X sprite 5
53259	Registro de posición Y sprite 5
53260	Registro de posición X sprite 6
53261	Registro de posición Y sprite 6
53262	Registro de posición X sprite 7
53263	Registro de posición Y sprite 7
53264	Registro de octavo bit

POKE53264,1:POKE53248,290–256*INT(290/256):POKE53249,70
En general para posicionar un sprite en los 343 puntos horizontales debemos hacer:

XA=INT(X/256):XB=X–256*XA
POKE53248,XB
POKE53248,1:REM SOLO SI XA 0.

donde X es la posición horizontal del sprite (en este caso tomamos al sprite 0).

Una vez ubicado en la pantalla, procedemos a activarlo, lo cual se logra a través del registro ubicado en la dirección 53269.
Cada bit de este registro indica qué sprite se activará. Si por ejemplo, el bit 0 está a "1", entonces se activará el sprite 0, si el bit 1 está a "1" se activará el sprite 1, si el bit 2 está en "1" se activará el 2, etc.
Si deseamos ampliar un sprite, se debe acceder a los registros correspondientes. En la dirección 53277 se encuentra el encargado de expandir el sprite horizontalmente mientras que en la dirección 53271 está el que lo expande verticalmente.
Al igual que el registro anterior, un "1" en 1:la posición correspondiente producirá la expansión.
Por ejemplo, si queremos expandir el sprite 0 en forma vertical y horizontal:
POKE53277,1:POKE53171,1
Para volverlo "normal":
POKE53277,0:POKE53271,0
Resumiendo, los procesos necesarios para trabajar con sprite son:
1) Definirlo en un área de memoria libre (usen el buffer del cassette y a partir de la dirección 49152)
2) Definir el color o los colores.
3) Ubicarlo en la pantalla (ojo si X >256).
4) Activarlo.
Finalmente les dejamos el programa correspondiente al listado 1, el cual se encarga de definir un sprite a través de "I". De esta manera se evitarán tener que realizar cuentas molestas.
El programa define una nave espacial y la imprime en pantalla.

Nota: ¡clr! significa que deben oprimir la tecla de SHIFT y HOME al mismo tiempo.

```
10 REM EDITOR DE SPRITES
12 P=0
20 FORT=0TO19
30 READA$:IFLEN(A$)>24THENPRINT"STRING
   DEMASIADO LARGO":STOP
40 FORJ=1TO3
50 FORI=7TO0STEP-1
60 C$=MID$(A$,J*8-I,1)
70 IFC$<>"."ANDC$<>"I"THENPRINT"CARACTERES
   NO DEFINIDOS":STOP
80 IFC$="."THEN100
90 V=V+2↑I
100 NEXTI
110 POKE896+P,V:P=P+1
120 V=0:NEXTJ
130 NEXTT
135 PRINT"!clr!":POKE53264,0
136 POKE2040,14:REM UBICACACION SPRITE 0
140 POKE53287,1:REM COLOR
150 POKE53248,120:REM POSICION X
160 POKE53249,120:REM POSICION Y
170 POKE53269,1:REM LO ACTIVAMOS
180 FORT=1TO4:REM LO MOVEMOS
190 FORX=1TO400
200 XA=INT(X/256)
210 XB=X-256*XA
220 POKE53248,XB
230 IFXA<>0THENPOKE53264,1
240 NEXTX
250 NEXTT:REM FINALIZA MOVIMIENTO
260 POKE53269,0:REM LO APAGAMOS
270 POKE53248,100:REM LO REUBICAMOS X
280 POKE53249,200:REM LO REUBICAMOS Y
290 POKE53264,0:REM DESACTIVAMOS OCTAVO BIT
300 FORD=1TO300:NEXT
310 POKE53277,1:FORD=1TO500:NEXT:REM LO EXPANDIMOS EN X
320 POKE53271,1:FORD=1TO500:NEXT:REM LO EXPANDIMOS EN Y
330 POKE53271,0:POKE53277,0:REM VOLVEMOS A LA NORMALIDAD
340 STOP
1000 REM CADA DATA TIENE 24 CARACTERES
1001 DATA.......IIIIIIIII........
1002 DATA.......IIIIIIIII........
1003 DATA.......II.....II........
1004 DATA.......II.....II........
1005 DATA.......II.....II........
1006 DATA.......II.....II........
1007 DATA.......II.....II........
1008 DATA.......IIIIIIIII........
1009 DATA.......IIIIIIIII........
1010 DATA.....IIIIIIIIIIII.......
1011 DATA.....II........II.......
1012 DATA..IIIII........IIIII....
1013 DATA..II..............II....
1014 DATA..II..............II....
1015 DATA..IIIIIIIIIIIIIIIIII....
1016 DATA..IIIIIIIIIIIIIIIIII....
1017 DATA.......II.....II........
1018 DATA.......II.....II........
1019 DATA.......II.....II........
1020 DATA.......II.....II........
```

MULTIPANTALLA PARA LA 64

```
20 PRINTTAB(10)"QQQQQQQQQQ ESTA ES LA PAN
TALA 1"
30 SYSAD+6,0,1
35 PRINTCHR$(147)
40 PRINTTAB(10)"QQQQQQQQQQ ESTA ES LA PAN
TALA 2"
50 SYSAD+6,0,2
55 PRINTCHR$(147)
60 PRINTTAB(10)"QQQQQQQQQQ ESTA ES LA PAN
TALA 3"
70 SYSAD+6,0,3
75 PRINTCHR$(147)
80 PRINTTAB(10)"QQQQQQQQQQ ESTA ES LA PAN
TALA 4"
90 SYSAD+6,0,4
91 PRINTCHR$(147)
92 PRINTTAB(10)"QQQQQQQQQQ ESTA ES LA PAN
TALA 5"
94 SYSAD+6,0,5
```

Tipo: Utilitario
Comp.: DC64
Conf: C-64 y/o Drive 1541

Este soft les permitirá definir cinco pantallas, a las cuales se puede acceder a través de comandos que más adelante explicaremos.
Podrán aplicarlo para desarrollar juegos y/o utilitarios.
El programa principal, escrito íntegramente en lenguaje máquina, corresponde al listado 1.
El listado 2 es un ejemplo muy sencillo de cómo aplicarlo.
Básicamente se definen cinco áreas o "slots" cuya longitud es igual al del área reservada para pantalla.

Los comandos orientados al manejo de cada slot son cuatro, y cada uno de ellos utiliza la instrucción SYS.
Por ejemplo, si queremos almacenar la pantalla actual en un determinado slot, se utiliza el comando:
SYS AD+6,0,NRO
donde NRO representa el número del slot (recuerden que debe ser un número comprendido entre 1 y 5).
Para imprimir un slot en pantalla se usa el comando:
SYS AD+6, 1,NRO
Además de almacenar y recuperar pantallas desde un slot, podemos guardarlas en el disco para más tarde recuperarlas.
Para ello se utiliza el comando:
SYS AD+3, NRO, "NOMBRE"
que carga al slot dado por NRO, la pantalla identificada con NOMBRE.
Si, en cambio, queremos almacenar un slot en disco, debemos utilizar:
SYS AD,NRO,"NOMBRE"
A través de NOMBRE identificamos en el disco la pantalla deseada.
Otra posible aplicación de este programa puede ser un scrolling, es decir ir trasladando toda la pantalla hacia la derecha o hacia la izquierda caracter por caracter.

Listado 1

```
10 REM MULTIPANTALLA
15 AD=51715
20 FORZ=51712TO52058:READY:C=C+Y:POKEZ,Y:NEXT
25 IFC<>45016THENPRINT"ERROR EN DATAS. VERIFIQUE LOS
   VALORES":STOP
30 PRINT"VALORES OK. GRABE EL PROGRAMA  Y LUEGO
   EFECTUE NEW"
50 :
60 REM SYSAD,SLOT#,"NOMBRE":GRABA SLOT EN DISCO.
70 REM SYS AD+3,SLOT#,"NOMBRE":CARGA SLOT DESDE
   EL DISCO.
80 :
90 REM SYSAD+6,0,SLOT#:GRABA PANTALLA EN EL SLOT
100 REM SYS AD+6,1,SLOT#:IMPRIME SLOT EN PANTALLA
110 :
120 REM EL NUMERO DE SLOT VA DESDE 1 A 5
130 :
140 REM EL PROGRAMA OCUPA
150 REM DESDE LA DIRECCION
160 REM $CA00 A LA $CBFF
170 REM NO INTERFIERE CON DOS 5.1
180 :
190 REM EL CUAL UTILIZA DESDE LAS
200 REM DIRECCIONES $C000 A $C800
210 REM
220 REM
230 DATA 76,168,202, 76,182,202, 76,243
240 DATA 202, 32,253,174, 32,235,183,165
```

```
250 DATA  20,133,  2,224,  0,240, 38,224
260 DATA   6,176, 34,202,188, 80,203,189
270 DATA  75,203,170,165,  2,208, 23,134
280 DATA 253,132,254,169,  0,133,251,169
290 DATA 216,133,252, 32, 90,202,169,  4
300 DATA 133,252, 32, 90,202, 96, 32, 35
310 DATA 203,134,251,132,252,169,216,133
320 DATA 254,169,  0,133,253, 32, 90,202
330 DATA 169,  4,133,254, 32, 90,202, 76
340 DATA  48,203,160,  0,162,  3,177,251
350 DATA 145,253,200,208,249,230,252,230
360 DATA 254,202,208,242,177,251,145,253
370 DATA 200,192,232,208,247,230,252,230
380 DATA 254, 96,147, 13, 13, 67, 79, 80
390 DATA  89, 82, 73, 71, 72, 84, 32, 49
400 DATA  57, 56, 52, 13, 76, 79, 85, 73
410 DATA  83, 32, 87, 65, 76, 76, 65, 67
420 DATA  69, 32, 38, 13, 75, 69, 78, 32
430 DATA  70, 82, 69, 78, 67, 72, 13,  0
440 DATA 162,  0,189,122,202,240,  6, 32
450 DATA 210,255,232,208,245, 96, 32,253
460 DATA 174, 32,158,183,224,  0,240, 48
470 DATA 224,  6,176, 44,202,188, 80,203
480 DATA 132,252,188, 75,203,132,251,188
490 DATA  85,203,132,254,169,232,133,253
500 DATA  32, 61,203, 32,189,255, 32, 35
510 DATA 203,169,  8,170, 32,186,255,169
520 DATA 251,166,253,164,254, 32,216,255
530 DATA  76, 48,203, 32,253,174, 32,158
540 DATA 183,224,  0,240, 37,224,  6,176
550 DATA  33,142, 90,203,169,  8,170,160
560 DATA   0, 32,186,255, 32, 61,203, 32
570 DATA 189,255,174, 90,203,202,188, 80
580 DATA 203,189, 75,203,170,169,  0, 32
590 DATA 213,255, 96,165,  0,  9,  1,133
600 DATA   0,165,  1, 41,254,133,  1, 96
610 DATA 165,  0,  9,  1,133,  0,165,  1
620 DATA   9,  1,133,  1, 96, 32,253,174
630 DATA  32,158,173, 32,130,183,166, 34
640 DATA 164, 35, 96,  0,  0,  0,  0,  0
650 DATA 160,168,176,184,192,167,175,183
660 DATA 191,199,  0
```

Listado 2

```
5 AD=51715
10 PRINTCHR$(147)
20 PRINTTAB(10)"!9 cr ab! ESTA ES LA PANTALA 1"
30 SYSAD+6,0,1
35 PRINTCHR$(147)
40 PRINTTAB(10)"!9 cr ab! ESTA ES LA PANTALA 2"
50 SYSAD+6,0,2
55 PRINTCHR$(147)
60 PRINTTAB(10)"!9 cr ab! ESTA ES LA PANTALA 3"
70 SYSAD+6,0,3
75 PRINTCHR$(147)
80 PRINTTAB(10)"!9 cr ab! ESTA ES LA PANTALA 4"
90 SYSAD+6,0,4
91 PRINTCHR$(147)
92 PRINTTAB(10)"!9 cr ab! ESTA ES LA PANTALA 5"
94 SYSAD+6,0,5
100 PRINTCHR$(147)
120 FORI=1TO5
130 SYS AD+6,1,I
140 FORI=1TO500:NEXT
150 NEXT
160 STOP
```

MANEJO DE ARCHIVOS (2ª Parte)

Continuamos describiendo el procedimiento para trabajar con archivos en los equipos Drean Commodore 16 y 64. En este número explicaremos el acceso relativo.



En el número anterior hemos comenzado la primera parte de esta nota hablando sobre las ventajas existentes cuando se trabaja con archivos.

Hemos visto la organización secuencial en donde el acceso a un determinado dato depende de los anteriores. Esto es, para acceder a uno en particular debemos pasar por los anteriores a él.

ACCESO RELATIVO

A diferencia del secuencial, en este tipo de acceso no hace falta pasar por los anteriores para tomar un dato en particular. Aquí, directamente, se accede al dato buscado. Cada uno de ellos tiene una posición relativa dentro del archivo. De esta manera existe el registro número uno, el dos, el tres, etc.

Así si nosotros queremos acceder a un registro, sólo debemos especificar la posición que ocupa dentro del archivo.

La figura 1 representa la organización del archivo DATO. Cada dato está representado por la posición dentro de él.

La C-64, al igual que la C-16, permite al usuario acceder a la información con solo especificar la posición en donde ella se encuentra.

Lamentablemente ambas máquinas no disponen de comandos orientadores a un cómodo manejo de datos. Se deben hacer una serie de preparativos antes de grabar o tomar registros desde el disco.

El procedimiento para poder escribir o leer información desde un archivo relativo es el siguiente:

1) Abrir el canal de comandos (número 15)
2) Abrir el canal de datos
3) Posicionarse sobre el registro en cuestión
4) Escribir o leer el registro

El primer punto se hace a través del tradicional OPEN. La disketera usa el canal número 15 para recibir los comandos provenientes desde la consola.

Como nosotros vamos a utilizar el comando "P", que nos permite posicionarnos sobre cualquier registro, debemos enviarlo por dicho canal. Seguidamente usamos los comandos INPUT o PRINT según sea el caso (ingresar o escribir datos).

El punto número dos se hace de la siguiente manera:

OPEN NRO , DI, CA, "Nombre,L," +CHR$(LO)

donde NRO representa el número de archivo (entre 1 y 255), DI

Figura 1

ARCHIVO
DATO

(1)	DATO A	Longitud del registro (1)=LEN(DATO A)
(2)	DATO B	Longitud del registro (2)=LEN(DATO B)
(3)	.	.
.	.	.
.	.	.
.	.	.
.	.	
(n)	DATO N	Longitud del registro (n)=LEN (DATO N)

representa el dispositivo en cuestión (para nuestro caso este será el número 8-disketera), CA es el canal (entre 1 y 15), Nombre indica el nombre del archivo y LO representa la longitud máxima de los registros (número de caracteres máximo).

Traten que la longitud (LO) sea un múltiplo de 256 (64,128, etc.). Esto se debe a que el sistema operativo del disco divide 256 (el tamaño de un bloque de disco) por la longitud aquí puesta.

Una advertencia importante: la L debe acompañar siempre al nombre del archivo.

Un ejemplo podrán verlo en el listado 1. Aquí abrimos un archivo relativo denominado DATO e indicamos a la disketera la longitud máxima de los registros.

Seguimos explicando otro comando fundamental para la utilización de archivos relativos, el **comando "P"**.

Este se debe enviar a la disketera por el canal de comandos. Su formato es el siguiente:

PRINT#1, "P"CHR$(CA+96)CRH$(REB) CHR$(REA) CHR$(PO)

donde CA representa el canal antes abierto (no hagan la suma), REB representa la parte baja del número del registro, REA indica la parte alta del número de registro y P la posición dentro del registro. Si este último se omite, la sistema asume que nos posicionamos a partir del primer caracter.

Ustedes se preguntarán por qué se necesita la parte alta y baja del número de registro o, algunos, se preguntarán qué es la parte alta y baja de un número.

Como la cantidad de registros que permite la 1541 trabajando con archivos relativos es de 700, no basta un byte para representarlo.

Como cada CHR$ puede tomar valores desde 1 a 256, se necesita un segundo CHR$. Dado un número cualquiera, las ecuaciones E1 y E2 lo convierten en su equivalente bajo-alto.

REA = INT(RE/256) –E1–
REB = RE – 256*REA–E2–

Observen que si el registro es menor a 256, la parte alta es siempre cero (0) y la parte baja es, directamente, el número del registro.

La ecuación E3 hace el pasaje inverso. Dado un número en formato bajo-alto ésta devuelve el número como lo conocemos.

RE = REB + 256*REA–E3– .

Dada la teoría necesaria, veamos un pequeño ejemplo de aplicación. El listado 2 es un ejemplo de cómo se abre un archivo relativo y cómo se graba un vector de 100 elementos.

Como tendremos 100 registros enumerados del 1 al 100 haremos, directamente, la parte alta REA a cero (0) y la parte baja REB la igualaremos al número de registro.

Desde la línea 20 hasta la 50 ponemos los valores del vector.

En la línea 60 realizamos la apertura del canal de comandos.

En la línea 70 abrimos el archivo de datos propiamente dicho e indicamos al sistema que la longitud máxima de los registros será de 64 caracteres (en realidad la máxima será de 2 caracteres).

En la línea 80 comenzamos el loop. En la línea 90 enviamos el comando de posicionamiento. De esta manera nos vamos posicionando de acuerdo al valor que va tomando I.

En la línea 100 escribimos el elementos V(I). Finalmente culminamos el lazo y cerramos los archivos.

Ustedes se preguntarán por qué no usar un archivo secuencial en vez de hacer tanto "lío". Bueno, la ventaja es que, ahora, cuando querramos tomar un dato, sólo bastará con indicar la posición.

El listado 3 realiza el proceso inverso que el listado 2. Aquí se pide el número del elemento y se lee desde el archivo.

En la línea 10 pedimos el ingreso del elemento buscado. En las líneas 20 y 30 realizamos las aperturas de los archivos.

En la línea 40 nos posicionamos sobre ese registro. Luego, en la línea 50 lo tomamos a través de la sentencia INPUT .

Finalmente imprimimos el dato y cerramos todos los archivos.

Para trabajar con este tipo de archivos hay que tener mucho cuidado. No olviden posicionarse sobre el registro antes de escribir o de leer.

Listado 1

```
OPEN 2,8,8,"DATO,L,"+CHR$(128)
```

Listado 2

```
10 REM EJEMPLO ARCHIVOS RELATIVOS
20 DIM V(100)
30 FORI=1TO100
40 V(I)=I
50 NEXTI
60 OPEN1,8,15
70 OPEN2,8,2,"DATOS,L,"+CHR$(64)
80 FORI=1TO100
90 PRINT#1,"P"CHR$(2+96)CHR$(I)
   CHR$(0)
100 PRINT#2,V(I)
110 NEXTI
120 CLOSE2
130 CLOSE1
140 END
```

Listado 3

```
10 INPUT"INGRESE EL NUMERO
   DEL ELEMENTO BUSCADO";E
20 OPEN1,8,15
30 OPEN2,8,2,"DATO,L,"+CHR$(64)
40 PRINT#1,"P"CHR$(2+96)CHR$(E)
   CHR$(0)CHR$(1)
50 INPUT#2,DA
60 PRINT"EL DATO CORRESPONDIENTE
   A LA POSICION";E;"ES";DA
70 CLOSE2
80 CLOSE1
90 END
```

LOS REGISTROS DEL CHIP TED

En el número anterior hemos comenzado a describir información inédita sobre esta computadora. Ahora explicaremos los registros del circuito integrado más importante del equipo



Como hemos dicho en el número anterior el chip TED es el dispositivo más importante de la Drean Commodore 16. Básicamente se encarga del manejo de gráficos, sonido y comunicación con el exterior.

La Drean Commodore 64 dispone de tres circuitos integrados que manejan, en forma independiente, el sonido, los gráficos y las comunicaciones con el exterior.

Como ustedes saben, la 16 no dispone de una "port" para el usuario. Tal vez la implementación de la misma no se justificaba debido a las características del equipo.

De todas maneras se pueden hacer cosas bastante interesantes, sólo si sabemos cómo son y dónde se pueden ubicar los registros del TED.

Debido a la cantidad de funciones que debe realizar, el TED dispone de un total de 63 registros, cada uno de los cuales realiza una determinada función.

El acceso a ellos puede ser a través del Basic (usando las instrucciones POKE y PEEK) o a través del Assembler usando el monitor residente.

Comenzaremos explicando desde los registros 0 al 5. Pero antes realizaremos una pequeña explicación acerca de cómo trabaja el sistema.

En determinado tiempo el sistema debe realizar una interrupción en lo que estaba haciendo y efectuar determinadas tareas.

Alguna de esas tareas es comprobar si la tecla STOP ha sido presionada, leer las teclas que se opriman para luego imprimirlas en la pantalla e incrementar el reloj de software (TI). Se realizan en forma periódica, es decir que cada vez que pasa cierto tiempo comienza a ejecutarse.

Para ello se necesita un circuito que le indique al sistema cuando se llegue al tiempo en que se deban ejecutar esas tareas. El sistema utiliza un contador que se carga con un valor inicial. Luego de haber cargado dicho valor se procede a poner en marcha el contador que irá disminuyendo en uno hasta llegar a cero.

El funcionamiento de ese contador es independiente al sistema, es decir que no hace falta tomar el control del contador y disminuir su contenido en uno. Para que se entienda mejor, "lo hace solito".

Mientras el contador "cuenta para abajo" el sistema hace lo suyo, por ejemplo ejecuta un programa que el usuario pidió a través de un RUN.

Cuando el contador llegue a cero provocará la atención del sistema. El guardará todo lo que tienen sus registros en el stack (memoria auxiliar) e irá a ejecutar las tareas que antes mencionamos.

A todo esto el contador arrancó de nuevo con el valor inicial antes cargado. Cuando el sistema haya realizado todo esto retomará la ejecución del programa principal.

Los registros 0, 1, 2, 3, 4 y 5 representan los tres contadores de la Drean Commodore 16. Cada registro está asociado con una dirección de memoria (RAM). Este es el motivo por el cual podemos acceder a ellos.

Al ser direcciones de memoria, los valores que se pueden poner están comprendidos entre 0 y 255. Como los contadores pueden contar mucho más que

este valor, se usan dos bytes para indicar la parte baja y alta del contador.

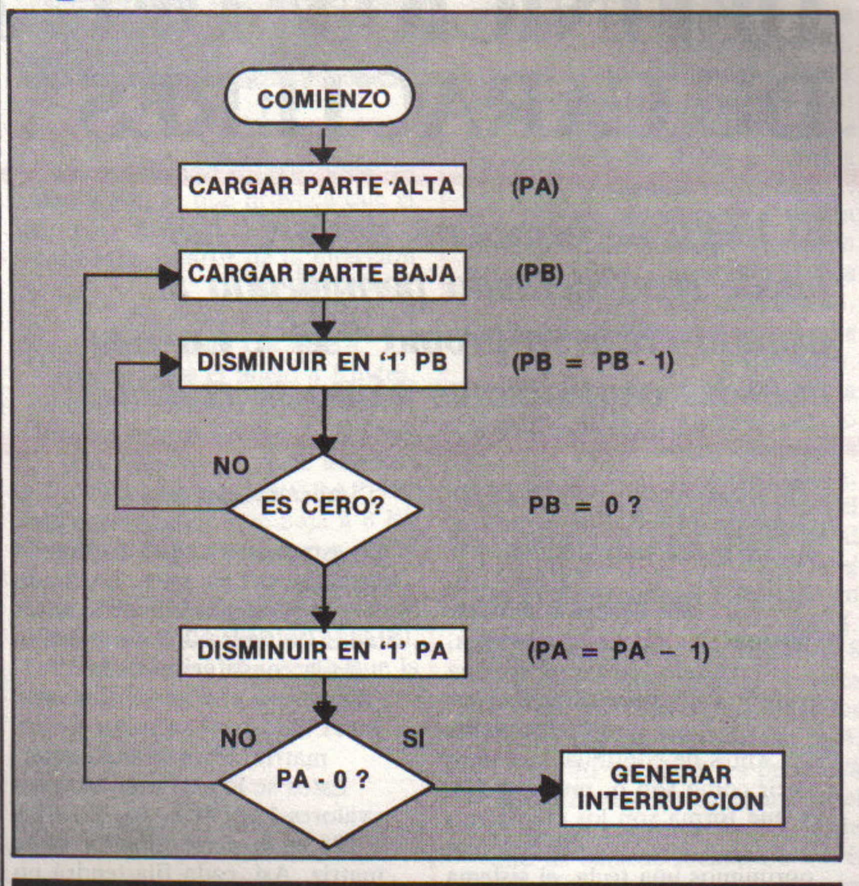
Así, entonces, primero se comienza a disminuir la parte baja. Cuando éste llegue a cero se disminuye en uno la parte alta. Si se llegó a cero se provoca la interrupción. Si no, se vuelve a cargar la parte baja con el valor original y se repite la operación.

La figura 1 muestra el diagrama de flujo correspondiente.

Los contadores disminuyen en uno a una velocidad de 884 KHz (debido al sistema de TV PAL). Es decir que cada 1.13 micro segundos (1 micro seg = 0.000001 seg aproximadamente) se disminuye el contador en uno. Por ejemplo, si la parte alta se carga con 255 ($FF) y la parte baja también con 255 ($FF), se realizará una interrupción recién cuando lleguen a cero, es decir a los 73.55 mili segundos (1 mili seg = 0,001 segundos).

Las direcciones en dichos contadores, comúnmente llamados timers, es la 65280 ($FF00 parte baja contador nro. 1), 65281 ($FF01 parte alta contador nro. 1), 65282 ($FF02 parte baja contador nro. 2), 65283 ($FF03 parte alta contador nro. 2), 65284 ($FF04 parte baja còntador nro. 3) y 65285 ($FF05 parte alta contador nro. 3) correspondientes a los registros 0, 1, 2, 3, 4 y 5. El timer 1 es utilizado por el sistema provocando la interrupción cada 16.66 mili segundos (ms) aproximadamente. Es decir que

Figura 1



no podemos usarlo.

Los timers 2 y 3 son contadores cargados siempre con 255 ($FF). De esta manera contabilizan tiempos de 73 ms. Prueben realizar desde el Basic la siguiente experiencia:

```
10 PRINT PEEK (65280) +
256*PEEK (65281)
20 GOTO 10
```

Verán cómo los valores impresos van disminuyendo y como vuelve a tomar los valores iniciales. Lo mismo pueden hacer con los timers 2 y 3. Aquí verán cómo los valores son muy superiores al contador anterior.

En el próximo número continuaremos explicando los registros del TED. El tema de los timers lo volveremos a tratar más adelante.

DEFINICION DEL JUEGO DE CARACTERES

El juego de caracteres de la C-128 puede ser redefinido permitiendo al usuario crear su propio "set". En esta nota les comentamos cómo se realiza.

Al igual que la C-64, la C-128 dispone de un juego de caracteres el cual puede ser redefinido por el usuario.

Para ello debemos cambiar algunos valores en la memoria, logrando así que el sistema "mire" a nuestros caracteres en lugar de los estandard.

Antes de continuar debemos decir cómo son o, mejor dicho, de qué forma son los caracteres.

Cada vez que nosotros oprimimos una tecla, el sistema determina una dirección, dentro del área en donde se ubica el juego de caracteres, en donde se almacenan los valores que forman a ese caracter.

Una vez localizada la dirección, se van a buscar los valores para imprimirle en pantalla.

Cada uno de los caracterese se representa a través de una matriz de 8 × 8 "puntitos", generalmente llamados dots. Si uno de ellos está a "1", el dot correspondiente se activará mientras que con un "0" se apagará.

De esta manera se va formando la imagen del caracter. La figura 1 representa la matriz antes citada. La figura 2 es un ejemplo de la constitución de la "A". Claro que el sistema almacena otros datos que, a partir de esa matriz, forman el caracter.

Estos se logran poniendo los valores 1, 2, 4, 8, 16, 32, 64 y 128 en la parte superior de la matriz. Así, cada fila tendrá un valor, el cual será la suma de aquellos cuando solo esté un punto activado.

En la figura 3 damos un ejemplo de los valores correspondientes a un caracter que representa un "cuadradito".

En la primera fila sólo sumamos los valores 64, 32, 16, 8, 4. Desde la segunda fila hasta la quinta sólo sumamos los valores 64 y 4. En la sexta fila volvemos a sumar los valores 64, 32, 16, 8, 4.

Así obtenemos la serie de valores 124, 68, 68, 68, 68, 124, 0, 0 los cuales representan a nuestro caracter.

Una vez comprendido el procedimiento para redefinir los caracteres, tenemos que indicarle al sistema dónde está el nuevo set.

Esto se hace debido a que el juego de caracteres se encuentra almacenado en ROM, en donde no es posible escribir datos. Por ello debemos trasladarlo a RAM para luego sí modificarlo.

La dirección 2604 ($A2C) es utilizada para dos propósitos. El primero de ellos es indicarle al sistema dónde se encuentra la dirección inicial de la memoria correspondiente a la pantalla. El segundo, el que nos interesa a nosotros, indica la dirección inicial del set de caracteres.

Como toda "buena" dirección de memoria, solo puede almacenar valores comprendidos entre 0 y 255. ¿Entonces cómo hace para cumplir con esas dos misiones? Los cuatro bits más altos es decir el 7, 6, 5 y 4, si comenzamos a enumerar a partir del 0, indican la dirección inicial de la pantalla. Mientras, los cuatro más bajos (el 3, 2, 1 y 0) señalan la dirección del juego de caracteres.

Figura 1

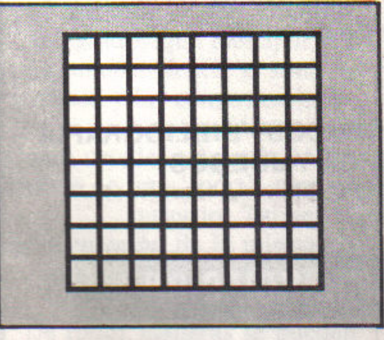


Figura 2

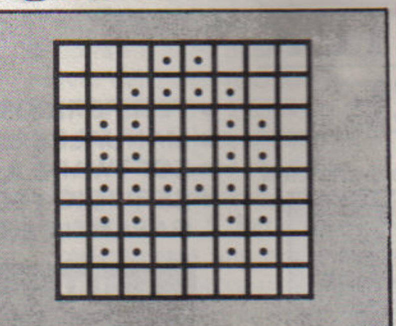
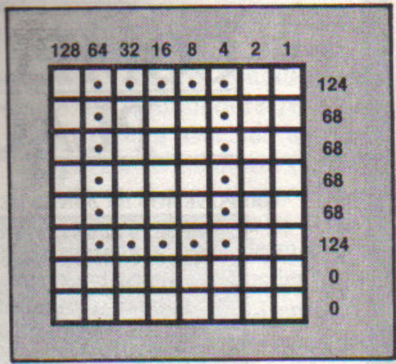


Figura 3



La figura 4 ejemplifica lo dicho.

Para determinar la dirección inicial, tanto de la pantalla como del set de caracteres, el sistema multiplica por 1024 a los valores respectivos.

Si ustedes realizan en modo directo PRINTPEEK(2604) obtendrán el valor 20 ($14). Es decir que la pantalla estará a partir de la dirección 1 × 1024 = 1024 (donde normalmente se ubica) y que el juego de caracteres comenzará a partir de la dirección 53248.

En este caso no se desprende tan fácilmente el inicio del juego de caracteres. El motivo de ello es que, en realidad, interviene dentro de la cuenta que indica la dirección inicial de éste, el banco de memoria.

Nosotros, para comprender mejor el procedimiento, omitiremos este parámetro.

Como dijimos anteriormente nuestros caracteres deben estar en RAM, es decir que debemos cambiar los cuatro bits más bajos para que ahora apunten a nuestro propio juego de caracteres.

Sólo nos queda decidir en qué lugar de la memoria pondremos nuestros caracteres. Esta zona debe ser una zona libre, es decir que no pueda ser borrada por programas Basic o por variables.

Comúnmente todos los programas Basic se almacenan, en la C-128, a partir de la dirección 7169 ($1C01). En caso de que se trabaje con gráficos en alta resolución el texto Basic se sube 8 Kb, lo que provoca que el área de almacenamiento comience a partir de la dirección 16384 ($4000).

Es decir que a través del comando GRAPHIC 2,1 levantaremos el inicio 8 Kbytes, permitiéndonos poner nuestro set de caracteres debajo de él y sin que nadie lo perturbe a menos que hagamos un GRAPHIC CLR, comando que baja a 8 Kb el texto Basic (lo vuelve a poner en su lugar).

Para ello pondremos en la dirección 2604 el valor 24 ($18), lo que le dirá al sistema que la pantalla comienza a partir de la dirección 1024 y que el set de caracteres se encuentra almacenado a partir de la dirección 8192.

Resumiendo, los pasos a seguir para modificar los caracteres son:

1) Trasladar el juego de caracteres desde ROM a RAM a través del listado 1.

2) Indicarle al sistema que la dirección inicial del set comienza a partir de la dirección 8192 a través de POKE2604,24.

3) Modificar los caracteres que se necesiten.

Una aplicación en concreto es la creación de la ñ, caracter que no se encuentra en ninguna computadora.

Para ello utilizaremos el programa correspondiente al listado 2, el cual traslada el juego de caracteres y lo pone en RAM.

Finalmente cambia los primeros 8 bytes (correspondiente al caracter "@") y pone en su lugar los valores que forman la ñ.

El listado 3 les permitirá visualizar en forma completa el set de caracteres de C-128. Cada uno de ellos se formará a través del "*" como punto activo y con un " " como punto apagado.

Figura 4

7 6 5 4 3 2 1 0

dirección inicial de pantalla — dirección inicial de juego de caracteres

Listado 1

```
10 FORI=0TO511
20 BANK14
30 A=PEEK(53248+I)
40 BANK0
50 POKE8192+I,P
60 NEXT
```

Listado 2

```
5 GRAPHIC 2,1
10 FORI=0TO511
20 BANK14
30 A=PEEK(53248+I)
40 BANK0
50 POKE8192+I,P
60 NEXT
70 BANK0:POKE2604,24
80 FORI=0TO7
90 READY
100 POKE8192+I,Y
110 NEXTI
120 DATA56,192,236,
252,220,204,204,0
```

Listado 3

```
10 BANK14
20 FORI=53248TO57343
30 A=PEEK(I)
40 FORX=7TO0STEP-1
50 IF(AAND(2↑X))=2↑X
   THENC$="*":GOTO70
60 C$=" "
70 PRINTC$;
80 NEXTX:PRINT
90 NEXTI:STOP
```

DEFINICION DE LAS TECLAS DE FUNCION

En el número anterior comenzamos a explicar cómo funciona la rutina IRQ y cuál es el procedimiento para ejecutar rutinas periódicamente. Ahora empezamos a describir los procedimientos necesarios para definir las teclas de función.



En el número anterior hemos desarrollado un programa que cambiaba el color del cursor.

Para ello utilizamos una rutina cuyo objetivo era incrementar el contenido de la dirección $287 (647), que representa el código de color del cursor.

A través del método antes explicado, modificamos los punteros de la rutina IRQ para que señalen nuestra rutina. De esta manera ella se estará ejecutando 60 veces por segundo.

Así no interferiremos con el sistema. Inclusive podemos desarrollar programas a medida que el cursor va cambiando de color.

No debemos olvidar lo más importante. Luego que nuestra rutina se ejecute debemos saltar a la rutina IRQ para que siga realizando las operaciones normales, como son el barrido del teclado, comprobación tecla stop, etc.

Finalmente haremos el desarrollo de proyecto original, es decir la definición de las teclas en función.

La idea es muy sencilla: cada vez que se oprima alguna de las teclas de función F1, F3, F5 o F7 debemos poner en el buffer del teclado los caracteres del comando deseado, juntamente con el código de RETURN para que se ejecute.

Luego ponemos el número de caracteres que se encuentran en el buffer en la dirección $C6 (198), que normalmente se utiliza para ello.

Hecho todo lo anterior saltamos a la IRQ normal para que tome los caracteres que nosotros pusimos y ejecute el comando.

Básicamente el programa está formado por tres módulos.

El primero (direcciones $C000-$C01F) se encarga de cambiar el puntero de la IRQ para que, ahora, señale a nuestra rutina.

Además inicializa los valores que luego se utilizarán.

El segundo módulo (direcciones $C021-$C06B), el más importante, es el que determina si se oprimió alguna de las teclas de función y, en base a ello, setea los parámetros requeridos por el

tercer módulo.
Si no se oprimió ninguna tecla de función, entonces se pone un cero en la dirección $C6 con lo cual se evita el "rebote" de las teclas.
El tercer módulo (direcciones $C070-$C084) se encarga de colocar en el buffer del teclado los caracteres que se encuentran a partir de la dirección de-

Listado 1

```
,C000 78          SEI
,C001 A9 21       LDA #$21
,C003 8D 14 03    STA $0314
,C006 A9 C0       LDA #$C0
,C008 8D 15 03    STA $0315
,C00B A9 04       LDA #$04
,C00D 8D 00 C1    STA $C100
,C010 A9 05       LDA #$05
,C012 8D 01 C1    STA $C101
,C015 A9 06       LDA #$06
,C017 8D 02 C1    STA $C102
,C01A A9 03       LDA #$03
,C01C 8D 03 C1    STA $C103
,C01F 58          CLI
,C020 60          RTS
,C021 A2 00       LDX #$00
,C023 BD 00 C1    LDA $C100,X
,C026 C5 C5       CMP $C5
,C028 F0 0C       BEQ $C036
,C02A E8          INX
,C02B E0 04       CPX #$04
,C02D D0 F4       BNE $C023
,C02F A9 00       LDA #$00
,C031 85 C6       STA $C6
,C033 4C 31 EA    JMP $EA31
,C036 E0 00       CPX #$00
,C038 D0 0B       BNE $C045
,C03A A9 00       LDA #$00
,C03C 85 14       STA $14
,C03E A9 C2       LDA #$C2
,C040 85 15       STA $15
,C042 4C 70 C0    JMP $C070
,C045 E0 01       CPX #$01
,C047 D0 0B       BNE $C054
,C049 A9 08       LDA #$08
,C04B 85 14       STA $14
,C04D A9 C2       LDA #$C2
,C04F 85 15       STA $15
,C051 4C 70 C0    JMP $C070
,C054 E0 02       CPX #$02
,C056 D0 0B       BNE $C063
,C058 A9 10       LDA #$10
,C05A 85 14       STA $14
,C05C A9 C2       LDA #$C2
,C05E 85 15       STA $15
,C060 4C 70 C0    JMP $C070
,C063 A9 18       LDA #$18
,C065 85 14       STA $14
,C067 A9 C2       LDA #$C2
,C069 85 15       STA $15
,C06B 4C 70 C0    JMP $C070
,C06E 00          BRK
,C06F 00          BRK
,C070 A0 00       LDY #$00
,C072 B1 14       LDA ($14),Y
,C074 C9 00       CMP #$00
,C076 F0 07       BEQ $C07F
,C078 99 77 02    STA $0277,Y
,C07B C8          INY
,C07C 4C 72 C0    JMP $C072
,C07F C8          INY
,C080 84 C6       STY $C6
,C082 4C 31 EA    JMP $EA31
```

vuelta en la dirección $14 y $15.
Esta finaliza cuando se encuentra con un cero (0). También pone la cantidad de caracteres leídos en $C6.
Después de ello salta a ejecutar la rutina IRQ normal.
El programa principal corresponde al listado 1 (para aquellos que lo quieran en Assembler) juntamente con su equivalente en Basic.
Aquellos que opten por este último, no olviden cargarlo en cinta o en disco una vez que lo hayan tipeado, ya que el programa se autoborra.
Por una cuestión de longitud del buffer y por ser ésta una primera versión, existen algunas limitaciones en lo que respecta a la longitud de los comandos o instrucciones.
Estas no deben superar los siete caracteres. Además, se debe poner un 0 al final del último caracter (es decir que en total se permiten hasta ocho caracteres).
De acuerdo a esto, el usuario debe POKEar en memoria cada caracter del comando respectivo.
Cada tecla de función dispone de un área de memoria donde nuestra rutina irá a buscar los caracteres cuando sea oprimida.
De esta manera determinamos las siguientes áreas:

Tecla	Dirección
F1	49664-49671
F3	49672-49679
F5	49680-49687
F7	49688-49695

El listado 3 es un programa que les permite definir las teclas de función. En este caso se pusieron LIST para F1, NEW con F3, PRINT con F5 y RUN con F7.
Lo deben ejecutar una vez que hayan activado el programa principal. Luego podrán borrarlo de la memoria, y los comandos seteados estarán activos en ella.
Les recomendamos que respeten las indicaciones en lo que respecta a longitud de comandos y finalización con cero. De otra manera existe una gran probabilidad de que la C-64 se cuelgue.
De todas maneras esperamos que esto haya servido para que ustedes profundicen más sobre este tema.
En próximos números continuaremos desarrollando temas, como el diseño de un reloj, usando la IRQ.

Listado 2

```
10 FORI=49152TO49284
20 READA:C=C+A
30 POKEI,A
40 NEXTI
50 IFC<>14999THENPRINT"ERROR EN DATAS. VERIFIQUE
   VALORES":STOP
60 SYS49152
70 PRINT"TECLAS DE FUNCION ACTIVADAS"
80 NEW
90 REM VALORES CODIGO MAQUINA
100 DATA 120,169, 33,141, 20,  3,169,192,141, 21
110 DATA   3,169,  4,141,  0,193,169,  5,141,  1
120 DATA 193,169,  6,141,  2,193,169,  3,141,  3
130 DATA 193, 88, 96,162,  0,189,  0,193,197,197
140 DATA 240, 12,232,224,  4,208,244,169,  0,133
150 DATA 198, 76, 49,234,224,  0,208, 11,169,  0
160 DATA 133, 20,169,194,133, 21, 76,112,192,224
170 DATA   1,208, 11,169,  8,133, 20,169,194,133
180 DATA  21, 76,112,192,224,  2,208, 11,169, 16
190 DATA 133, 20,169,194,133, 21, 76,112,192,169
200 DATA  24,133, 20,169,194,133, 21, 76,112,192
210 DATA   0,  0,160,  0,177, 20,201,  0,240,  7
220 DATA 153,119,  2,200, 76,114,192,200,132,198
230 DATA  76, 49,234
```

Listado 3

```
10 C$(1)="LIST":M(1)=49664
20 C$(2)="NEW" :M(2)=49672
30 C$(3)="PRINT":M(3)=49680
40 C$(4)="RUN":M(4)=49688
50 FORI=1TO4
60 FORL=0TOLEN(C$(I))-1
70 POKEM(I)+L,ASC(MID$(C$(I),L+1,1))
80 NEXTL
90 POKEM(I)+L,13:POKEM(I)+L+1,0
100 NEXTI
```

COMO USAR BIEN EL BASIC

En esta nota exponemos algunos detalles importantes que deben tenerse en cuenta al momento de comenzar a programar. Para los neófitos y avanzados en computación.

Antes de comenzar a desarrollar un determinado programa se debe entender a la perfección cuál o cuáles son los objetivos del mismo.
Hecho esto no nos debemos desesperar y comenzar a tipear las líneas que formarán al programa sino que, por el contrario, debemos realizar una planificación del trabajo.
Para ello una de las herramientas que comúnmente se utiliza es el famoso "diagrama de flujo". Se lo usa para representar, a través de símbolos predefinidos, los distintos procesos que nuestro programa efectuará.
Rombos de decisión, líneas de flujo, rectángulos de proceso, son algunos de esos símbolos (Figura 1).
Otro de los métodos utilizados es el de "refinación sucesiva". Aquí se describen, en primera instancia, los procesos a realizar en una forma muy generalizada.
Luego, cada uno de los anteriores procesos se especifican un poco más. Así se logran subprocesos que luego se los divide nuevamente en otros más pequeños.
Veamos un ejemplo simple para que esto se comprenda mejor: Leer dos números e imprimir su suma.
El proceso básico se muestra en la figura 2. Formado por tres grandes procesos que son: Leer dos números, Sumarlos y Mostrar el resultado.
A partir de aquí iniciamos el refinamiento. El primer proceso (Leer dos números) lo especificamos aun más como Leer el primer número y Leer el segundo número.
En el segundo proceso (Sumarlos) especificamos que se debe tomar el primer número y sumarle el segundo.
En el último proceso (Mostrar el resultado) aclaramos que se debe imprimir la suma.
De esta manera hemos realizado la primera refinación. Como nosotros queremos un programa escrito en Basic que lleve a cabo este procedimiento, necesitamos seguir dividiendo cada uno de los subprocesos e instrucciones entendibles para la computadora.
Es así como se realiza el segundo refinamiento (figura 3). Leer el primer y segundo número significa que debemos utilizar la sentencia Basic INPUT.
Por ello ponemos INPUT A e INPUT B.
Tomar el primer número y sumarle el segundo se representa con C=A+B. Finalmente mostrar el resultado equivale a PRINT C.
Descartando el problema propuesto, este método es muy ventajoso cuando se deben desarrollar grandes programas.
Otro método bastante eficaz consiste en utilizar la herramienta humana más poderosa: la abstracción.
¿Cómo es esto? Para contestar explicaremos antes qué son las subrutinas y cómo podemos crearlas a partir de nuestra abstracción.
Una subrutina es un conjunto de instrucciones. Al ser común su utilización en distintas partes del programa se las define con un nombre y se accede a ellas en cualquier momento.
En el caso del Basic de los equipos Drean Commodore el acceso a éstas se realiza especificando el número de línea inicial de la misma dentro del programa.
Para definir una subrutina conviene imaginarla como una "caja negra" (nos abstraemos en lo que respecta al interior de esa caja) donde existe sólo una entrada y una salida.
De esta manera especificamos que ante un determinado tipo de entrada, la caja negra debe devolver una salida determinada.
Por ejemplo, si nuestro programa utiliza una subrutina que dado un número 'X' devuelve su cuadrado (X↑2). El esquema respectivo se representa en la figura 4.
Primero la caja negra. Luego los refinamientos sucesivos.

Figura 1

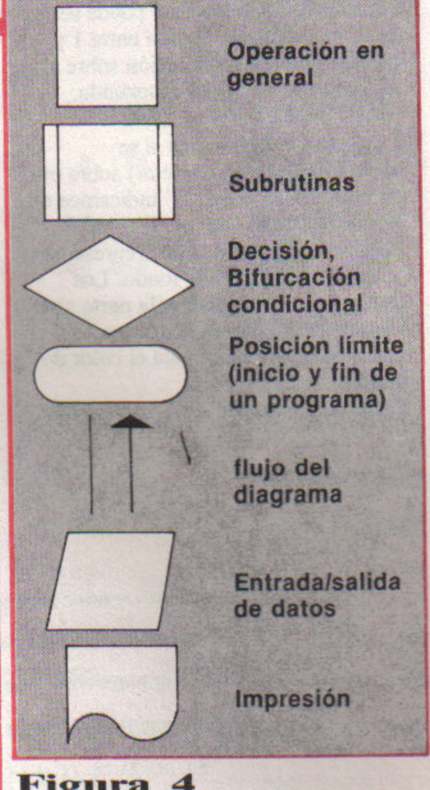
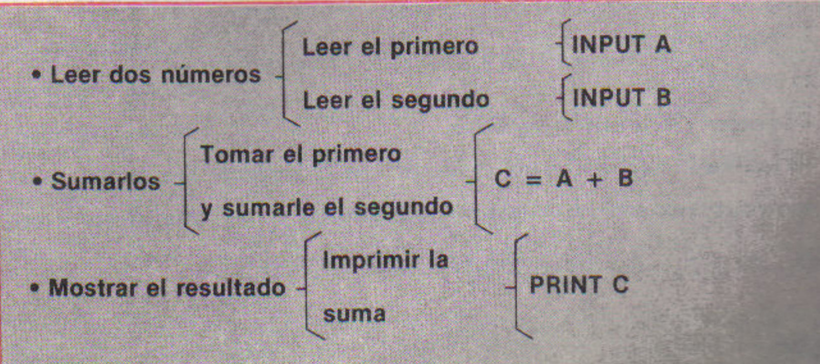


Figura 4



Figura 2-3



AMPLIADOR DE CARACTERES

Clase: Utilitario
Comp.: Drean Commodore 64
Conf.: Básica

El programa que aquí presentamos les permitirá aumentar e imprimir en alta resolución un caracter acorde a los parámetros ingresados.
Estos se refieren a las coordenadas en pantalla donde será impreso, el factor de ampliación sobre eje X e Y que sufrirá el caracter, el código ASCII del caracter a imprimir, la sobreimpresión, y color del caracter.
Cada uno de ellos debe ser ingresado en determinadas direcciones de memoria, para luego poder llamar a la rutina correspondiente.
Así se la podrá utilizar como una especie de subrutina, la cual imprimirá sobre la pantalla texto, indicaciones o instrucciones para juegos, programas, etc., con el formato definido por nosotros.
Las direcciones de memoria correspondientes a los distintos parámetros son las que se indican en la figura 1.
En las direcciones 678 y 679 se debe colocar las coordenadas en pantalla en donde será impreso el caracter, con valores comprendidos entre 0 y 39 para la primera (eje X); 0 y 24 para la segunda (eje Y).
En las direcciones 681 y 682 se debe ingresar el factor de aumento del caracter a imprimir. Esta puede ser un valor entero comprendido entre 1 y 40 para la primera (ampliación sobre el eje X); entre 1 y 25 para la segunda (ampliación eje Y).
La dirección 820 indica si se sobreimprimirá (el caracter) sobre otro ya impreso. Con un "1" indicamos que sí, mientras que con un "0", no.
La última dirección (820) representa el color del caracter y de fondo. Los cuatro bits más pesados (la parte alta) representa el color del caracter, mientras que la parte baja el color de fondo.
Conjuntamente con estas direcciones, se encuentran una serie de rutinas que realizan determinadas tareas. Estas se describen en la figura 2.
Carguen el programa correspondiente al listado 1 y almacénenlo antes de ejecutarlo, ya que se autoborra de la memoria.
El listado 2 es un ejemplo de la utilización de este utilitario. Observen como se puede utilizar como una subrutina.
Para proteger la pantalla de alta resolución y el mapa de color, seteen el límite de memoria a través de POKE55,0:POKE56,92.

Figura 1

Dir.	Descripción
678	Coordenada eje X (0-39)
679	Coordenada eje Y (0-24)
681	Ampliación sobre eje X
682	Ampliación sobre eje Y
683	Código ASCII del caracter a imprimir
820	Sobreimpreso (1=si/0=no)
821	Byte de color

Figura 2

SYS 32768: Prende la pantalla para trabajar en alta resolución.
SYS 32771: Limpia la pantalla llenándola con el color de fondo.
SYS 32774: Regresa a la pantalla original.
SYS 32777: Imprime el caracter.

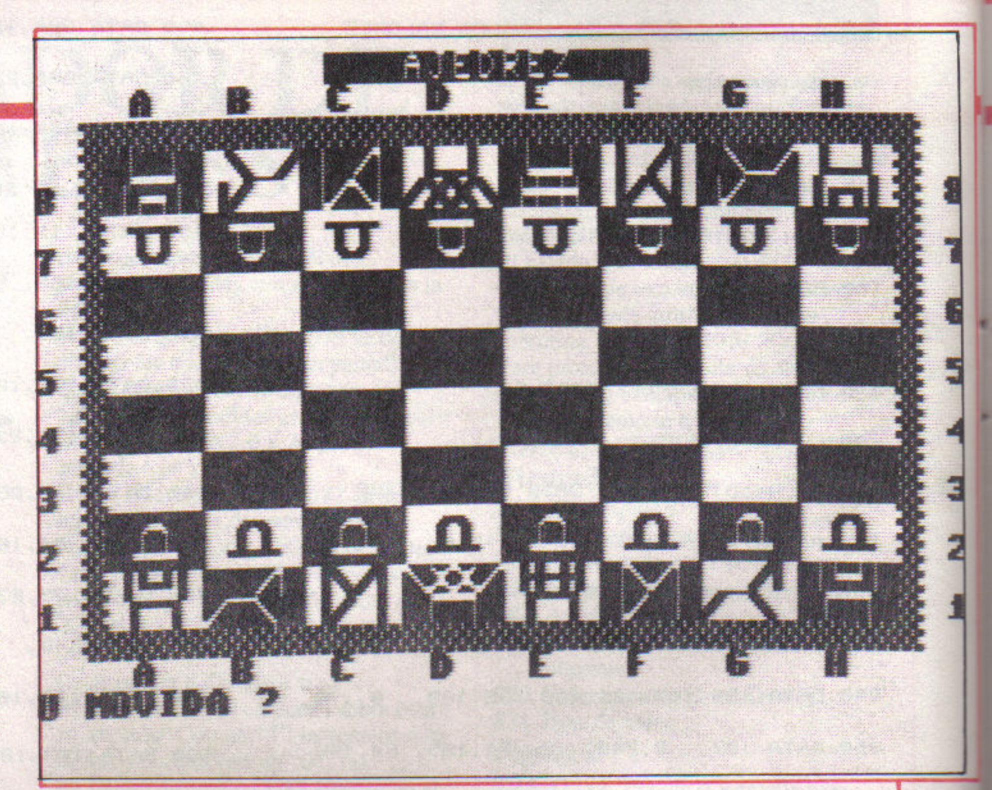
Listado 2

```
10 REM PRUEBA DE FUNCIONAMIENTO
20 REM DEL 'AMPLIADOR DE CARACTERES'
22 REM 53281:COLOR DE FONDO PANTALLA
30 REM 678:COOR. X
32 REM 679:COOR. Y
34 REM 681:AMPLIACION EJE X
36 REM 682:AMPLIACION EJE Y
38 REM 683:CODIGO ASCII DEL CARACTER
40 REM 820:SOBRE IMPRESO (1=SI/0=NO)
41 SYS32768:REM ACTIVA PANTALLA
42 REM 821:BYTE DE COLOR
43 POKE53281,0
44 SYS32771:REM LIMPIA PANTALLA
45 A$="AMPLIADOR":CX=13:CY=2:AX=1:AY=2:F=1:GOSUB50
46 A$="DE":CX=14:CY=6:AX=2:AY=4:F=2:GOSUB50
47 A$="CARACTERES":CX=1:CY=11:AX=4:AY=10:F=3:GOSUB50
48 A$="DREAN COMMODORE":CX=3:CY=22:AX=2:AY=1:F=2:
   GOSUB50:GOTO200
50 FORI=1TOLEN(A$)
52 POKE678,CX+F*I
54 POKE679,CY
56 POKE681,AX
58 POKE682,AY
60 POKE683,ASC(MID$(A$,I,1))
64 POKE820,0
66 POKE821,11
68 SYS32777:REM IMPRIME CARACTER
70 NEXTI:RETURN
200 GETA$:IFA$=""THEN200
210 SYS32774:REM VUELVE A PANTALLA INICIAL
```

Listado 1

```
120 PRINT"!clr!!cr ab!CARGANDO VALORES
   EN MEMORIA"
130 PRINT"!2cr ab!UN MOMENTO..."
140 READA,B,D
150 REM COMIENZA LECTURA VALORES
160 FORK=ATOB
170 READC:POKEK,C
180 POKE1024,C:POKE55296,C
190 CH=CH+C:NEXT
200 IFCH<>DTHENPRINT"DATA ERROR":STOP
210 PRINT"!2cr ab!DATOS OK":NEW
220 DATA32768,33223,47644
230 DATA  76, 95,129, 76,143,129, 76,119
240 DATA 129,169, 96,133, 35,169,  0,174
250 DATA 167,  2,240, 12, 24,105, 64,144
260 DATA   2,230, 35,230, 35,202,208,244
270 DATA 174,166,  2,240, 10, 24,105,  8
280 DATA 144,  2,230, 35,202,208,246,133
290 DATA  34,120,165,  1, 41,251,133,  1
300 DATA 169,  0,133, 21,173,171,  2,133
310 DATA  20,  6, 20, 38, 21,  6, 20, 38
320 DATA  21,  6, 20, 38, 21, 24,169,216
330 DATA 101, 2',!33, 21,160,  7,177, 20
340 DATA 153,174,  2,136, 16,248,165,  1
350 DATA*  9,  4,133,  1, 88,169,  0,141
360 DATA 187,  2,141,173,  2,173,170,  2
370 DATA 141,172,  2,173,173,  2,141,185
380 DATA   2,174,187,  2,160,  7,126,174
390 DATA   2,176,  3,169,  0, 44,169,255
400 DATA 153, 60,  3,136, 16,240,126,174
410 DATA   2,169,  0,141,182,  2,169,  8
420 DATA 141,186,  2,162,  0,172,169,  2
430 DATA 189, 60,  3,240,  7, 56, 46,182
440 DATA   2, 76,176,128, 24, 46,182,  2
450 DATA 206,186,  2,240, 11,136,208,232
460 DATA 232,224,  8,208,224, 76,241,128
470 DATA 140,183,  2,142,184,  2,172,185
480 DATA   2,173, 52,  3,240,  2,177, 34
490 DATA  13,182,  2,145, 34,169,  0,141
500 DATA 182,  2,169,  8,141,186,  2,173
510 DATA 185,  2, 24,105,  8,141,185,  2
520 DATA 172,183,  2,174,184,  2, 76,181
530 DATA 128,238,173,  2,173,173,  2,201
540 DATA   8,208, 18,169,  0,141,173,  2
550 DATA  24,165, 34,105, 64,144,  2,230
560 DATA  35,230, 35,133, 34,206,172,  2
570 DATA 240,  3, 76,115,128,238,187,  2
580 DATA 173,187,  2,201,  8,240,  3, 76
590 DATA 109,128,169, 92,133, 21,169,  0
600 DATA 174,167,  2,240, 10, 24,105, 40
610 DATA 144,  2,230, 21,202,208,246, 24
620 DATA 109,166,  2,133, 20,144,  2,230
630 DATA  21,174,170,  2,172,169,  2,136
640 DATA 173, 53,  3,145, 20,136, 16,251
650 DATA 165, 20, 24,105, 40,144,  2,230
660 DATA  21,133, 20,202,208,230, 96,173
670 DATA  17,208,  9, 32,141, 17,208,169
680 DATA 120,141, 24,208,173,  0,221, 41
690 DATA 252,  9,  2,141,  0,221, 96,173
700 DATA  17,208, 41,223,141, 17,208,169
710 DATA  21,141, 24,208,173,  0,221, 41
720 DATA 252,  9,  3,141,  0,221, 96,160
730 DATA   0,169, 96,133, 21,132, 20,162
740 DATA  32,169,  0,145, 20,136,208,251
750 DATA 230, 21,202,208,246,173, 33,208
760 DATA  41, 15,133,  2, 10, 10, 10, 10
770 DATA 160,  0,  5,  2,153,  0, 92,153
780 DATA   0, 93,153,  0, 94,153,232, 94
790 DATA 200,208,241, 96,  0,  0,  0,  0
```

AJEDREZ



Tipo: Juego
Comp.: DC64
Conf.: Básica

Este programa les permitirá a dos jugadores utilizar su Drean Commodore 64 como tablero de ajedrez.

El tablero se imprimirá con las fichas y también con las coordenadas de las respectivas posiciones.

Cada movimiento se lleva a cabo a través de la coordenada inicial y la coordenada final de la ficha en cuestión, es decir tipo A1A2, C3D5, etc.

Las primeras en mover son las blancas, luego les toca a las negras. La computadora pedirá la movida a través de SU MOVIMIENTO?

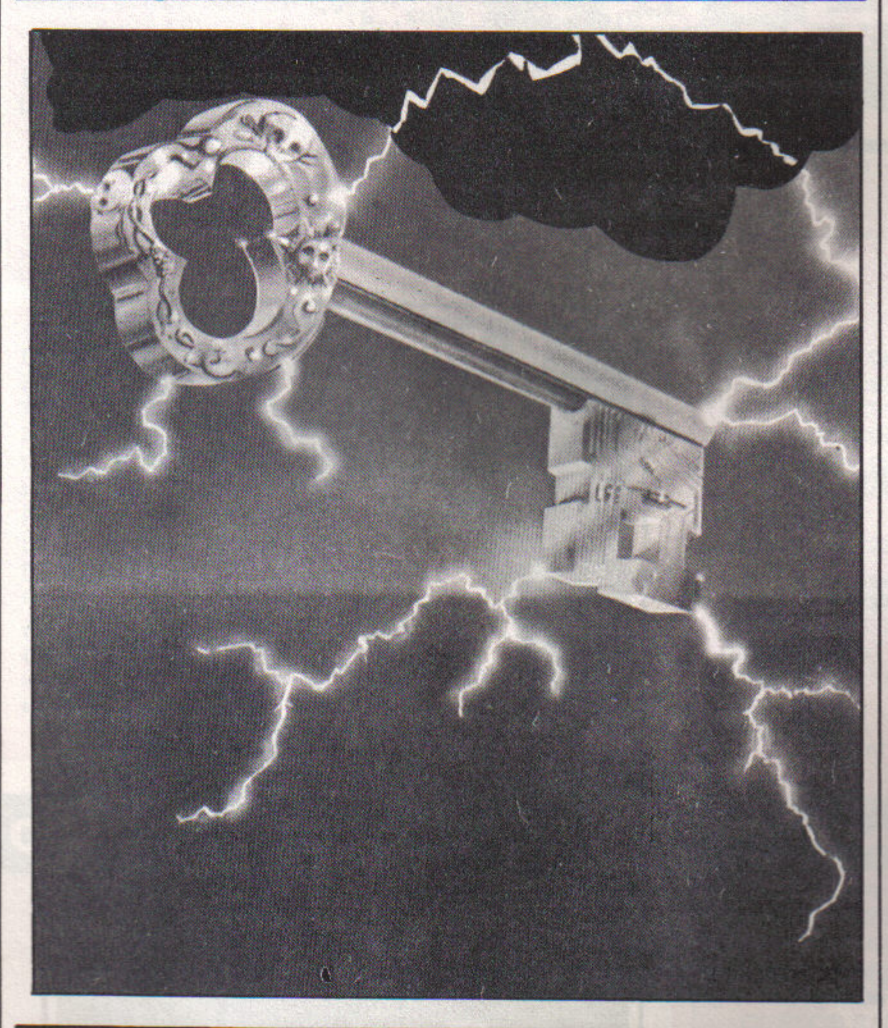
En caso de que ingresemos un formato de movimiento distinto, se imprimirá el mensaje correspondiente.

```
10 REM AJEDREZ
20 GOSUB20000
30 PRINT"▪";:POKE53280,1:POKE53281,1
1000 GOSUB9000
1100 GOSUB8500
1200 GOSUB11000
1300 L$=LEFT$(T$,2):GOSUB12000
1310 IFF<>0THEN1400
1320 PRINT:PRINT"NO HAY PIEZAS EN ESE LUGAR"
1330 FORI=1TO1000:NEXT:GOTO1200
1400 D$(F)=RIGHT$(T$,2)
1405 FORI=1TO10
1410 F$=LEFT$(T$,2):D=D(F)
1420 GOSUB8000
1430 F$=RIGHT$(T$,2):D=0
1440 GOSUB8000
1450 F$=LEFT$(T$,2):D=0
1460 GOSUB8000
1470 F$=RIGHT$(T$,2):D=D(F)
1480 GOSUB8000
1490 NEXT
1500 GOTO1200
7999 END
8000 REM PONE PIEZAS
8040 T=ABS(D):IFD<0THENT=T+9
8050 Q$=P$(T)
8080 X=ASC(F$) 64:Y=VAL(RIGHT$(F$,1))
8100 G=X+Y
8120 R$=""
```

```
8140 IFG/2<>INT(G/2)THENR$="█"
8160 PRINTLEFT$(D$,4+2*(8-Y));SPC(4+(4*(X-1)))
     ;R$;O$;"█"
8200 RETURN
8500 REM PONE TABLERO
8520 FORI=1TO34
8540 IFD$(I)=""THEN8600
8550 F$=D$(I):D=D(I)
8570 GOSUB8000
8600 NEXT
8650 RETURN
9000 PRINT"█            █  AJEDREZ   "
9003 PRINT"    A  B  C  D  E  F  G  H  "
9005 PRINT"  ████████████████████████████"
9010 FORI=1TO4
9020 FORJ=1TO2
9025 A$=" "
9030 IFJ=2THENA$=CHR$(58-2*I)
9040 PRINT" "A$" █";
9050 FORK=1TO4:PRINT"█   █   ";:NEXT
9060 PRINT"█ "A$
9070 NEXT
9080 FORJ=1TO2
9085 A$=" "
9090 IFJ=2THENA$=CHR$(57-2*I)
9100 PRINT" "A$" █";
9110 FORK=1TO4:PRINT"   █   █";:NEXT
9120 PRINT"█ "A$
9130 NEXT
9140 NEXT
9150 PRINT"  ████████████████████████████"
9155 PRINT"    A  B  C  D  E  F  G  H  "
9160 RETURN
10000 REM SE PREPARA MOVIDA
10010 PRINT"██";
10020 FORI=1TO50
10030 GETS$:IFS$<>""THEN10100
10040 NEXT
10050 PRINT" █";
10060 FORI=1TO50
10070 GETS$:IFS$<>""THEN10100
10080 NEXT
10090 GOTO10010
10100 PRINT" █";
10140 RETURN
11000 REM TOMA MOVIDA
11010 PRINTD$;E$:PRINTE$;D$"SU MOVIDA ? ";
11015 T$=""
11020 FORJ=1TO4
11030 GOSUB10000
11040 S=ASC(S$)
11050 IF(J=1ORJ=3)AND(S<65ORS>72)THEN11200
11060 IF(J=2ORJ=4)AND(S<49ORS>56)THEN11200
11080 PRINTS$;:T$=T$+S$
11090 NEXT
11100 PRINT" OK, (OPRIMA RETURN)"
11110 GOSUB10000
11120 IFS$<>CHR$(13)THEN11010
11180 RETURN
11200 PRINT:PRINT"INGRESO INCORRECTA";
11210 FORI=1TO1000:NEXT
11220 GOTO11010
12000 REM CHECK PIECE
12010 F=0
12020 FORI=1TO34
12030 IFD$(I)=L$THENF=I
12040 NEXT
12050 RETURN
20000 DIM P$(18),D(34),D$(34)
20010 FORI=0TO18:READ P$(I):NEXT
20020 FORI=1TO34:READ D(I),D$(I):NEXT
20030 D$="█":FORI=1TO21:D$=D$+"█":NEXT
20040 FORI=1TO39:E$=E$+" ":NEXT
20050 FORI=1TO39:E$=E$+"█":NEXT
21000 RETURN
30000 REM PIECE DATA
30005 DATA"    █████    "
30010 DATA" ┬┬ █████ ◡ "
30020 DATA" \_/█████|/  "
30030 DATA" |/|█████|\| "
30040 DATA""
30050 DATA" └┘ █████|┌| "
30060 DATA""
30070 DATA""
30080 DATA" └┘ █████/XX\"
30090 DATA" └┘ █████ H-H "
30110 DATA" ⌒ █████ ┴┴ "
30120 DATA"  /|█████/‾\ "
30130 DATA" |\/|█████|/| "
30140 DATA""
30150 DATA" |┘| █████ ┌┐ "
30160 DATA""
30170 DATA""
30180 DATA"\XX/█████ ┌┐ "
30190 DATA" H-H █████ ┌┐ "
31000 REM DATOS DEL TABLERO
31010 DATA  1,A7, 1,B7, 1,C7, 1,D7, 1,E7,
     1,F7, 1,G7, 1,H7
31020 DATA  2,B8, 2,G8
31030 DATA  3,C8, 3,F8
31050 DATA  5,A8, 5,H8
31080 DATA  8,D8, 8,""
31090 DATA  9,E8
31110 DATA -1,A2,-1,B2,-1,C2,-1,D2,-1,E2,
     -1,F2,-1,G2,-1,H2
31120 DATA -2,B1,-2,G1
31130 DATA -3,C1,-3,F1
31150 DATA -5,A1,-5,H1
31180 DATA -8,D1,-8,""
31190 DATA -9,E1
READY.
```

LAS SUBRUTINAS DEL DREAN COMMODORE 64

Continuamos explicando las rutinas que constituyen el sistema operativo de la C-64. En este número describimos, entre otras, la rutina IONIT.



Nombre de la función: IOBASE
Propósito: Indica la dirección base de los periféricos de entrada/salida
Dirección de llamada: $FFF3 (hex) 65523 (dec)
Registros de comunicación: X,Y
Rutina preliminar: Ninguna
Error: Ninguno
Requerimientos de stack: 2
Registros afectados: X,Y
Descripción: Esta rutina devuelve en los registros X e Y la dirección de inicio del área que se utiliza para operaciones de entrada/salida.

De acuerdo al formato utilizado en la C-64, primero se pone el byte bajo (registro X) y luego el byte alto (registro Y).

Si acceden a esta rutina comprobarán que al retornar de ella, el registro X contendrá el valor $00 y el registro Y tendrá $DC (220).

Esto indica que la dirección inicial del área de E/S es $DC00, dirección donde se comienzan a almacenar los registros de las dos CIA 6526.

Pasos a seguir:

1) Acceder a la rutina.
2) Leer los registros X e Y.
3) Almacenarlos en dos direcciones consecutivas de memoria en página cero.
4) Cargar el registro Y con el número de registro al cual se quiere acceder.

Ejemplo:

a) Desde el Assembler

JSR $FFF3: accedemos a la rutina
STX $14: almacenamos el X en $14
STY $15: almacenamos el Y en $15
LDY#$2: cargamos el Y con el 2 para acceder al DDRA (dirección $DC02)
LDA#$0: hacemos que en el Port A sean todos salida
STA ($14),Y: a través del direccionamiento indirecto indexado.

b) Desde el Basic

10 SYS 65523: REM accedemos a la rutina
20 BB=PEEK (781): REM leemos el byte bajo
30 BA=PEEK (782): REM leemos el byte alto
40 NR=2: REM seteamos el número de registro
50 A=0: REM preparamos el DORA como entrada
60 POKE (BA+256*BB)+NR,A: REM ponemos 0 en DC02
70 STOP

Nombre de la función: IOINIT
Propósito: Inicializa los periféricos de entrada/salida
Dirección de llamada: $FF84 (hex) 65412 (dec)
Registros de comunicación: Ninguno
Rutina preliminar: Ninguna
Error: Ninguno
Requerimientos de stack: Ninguno

KERNAL

Registros afectados: A,X,Y
Descripción: Esta rutina inicializa todos los periféricos de entrada/salida
Pasos a seguir:
1) Acceder a la rutina
Ejemplo:
a) Desde el Assembler
JSR $FF84: inicializa los periféricos de E/S
b) Desde el Basic
10 SYS65412
Nombre de la función: LISTEN
Propósito: Pone un periférico en modo "escucha"
Dirección de llamada: $FFB1 (hex) 65457 (dec)
Registros de comunicación: A
Rutina preliminar: Ninguna
Error: Ver READST
Requerimientos de stack: Ninguno
Registros afectados: A
Descripción: Esta rutina prepara a un periférico sobre el bus serie para que reciba datos.
Para ello se debe cargar en el acumulador el número de periférico seleccionado.
Pasos a seguir:
1) Cargar el acumulador con el número de periférico
2) Acceder a la rutina
Ejemplo:
a) Desde el Assembler
LDA# $08: ponemos la disketera
JSR $FFB1: en modo escucha
b) Desde el Basic
10 POKE780,8: REM disketera
20 SYS65457: REM en recepción de datos
Recuerden que las direcciones 780, 781 y 782 representan al acumulador, registro X y registro Y.
Nombre de la función: LOAD
Propósito: Carga RAM desde un periférico
Dirección de llamada: $FFD5 (hex) 65493 (dec)
Registros de comunicación: A, X, Y
Rutina preliminar: SETLFS, SETNAM
Error: 0, 4, 5, 8, 9
Requerimientos de stack: Ninguno
Registros afectados: A, X, Y
Descripción: Esta rutina carga directamente datos desde un periférico sobre el bus serie en la memoria RAM.
Si el contenido del acumulador es cero se procederá a efectuar la operación de carga mientras que con un uno se realizará una verificación de los datos cargados.
Si el periférico en cuestión se ha abierto con la dirección secundaria 0, los datos almacenados se cargarán a partir de la dirección indicada por los registros X e Y (parte baja y alta de la dirección).
Si, en cambio, la dirección secundaria es 1 o 2, los datos se cargarán a partir de la dirección especificada en el encabezamiento del disco o cinta.
Para acceder a la rutina LOAD debemos, antes que nada setear los parámetros del archivo o programa que se cargará a través de las rutinas SETLFS y SETNAM, las cuales explicaremos más adelante.
Pasos a seguir:
1) Acceder a las rutinas SETLFS y SETNAM
2) Indicar si vamos a cargar o a verificar a través del contenido del acumulador (0 y 1 respectivamente)
3) Acceder a esta rutina
Ejemplo:
El ejemplo de la utilización de esta rutina lo haremos luego de explicar SETLFS y SETNAM.

TRUCOS

Lectura del canal de error

**Como se sabe, el led rojo del Disk Drive 1541 titila cuando ocurre un error en la operación de éste.
El sistemá operativo de la unidad (DOS) pone en el canal de comandos (Nº 15) el mensaje de error correspondiente. Se puede leer haciendo:
10 OPEN15,8,15:INPUT15, CODIGO,MENSAJE$;TRACK, SECTOR:PRINT CODIGO, MENSAJES,TRACK,SECTOR: CLOSE15:STOP
Luego de efectuar RUN se imprimirá en la pantalla el número de error (CODIGO), el error (MENSAJE$), el TRACK y el SECTOR en donde ocurrió.
Estas sentencias no pueden ejecutarse en modo directo ya que la sentencia INPUT es válida solamente en un programa.
De todas maneras se sugiere que se consulte al manual de la 1541.**

Gráficos con la C-128

**Aquí hay un interesante programa que realiza un gráfico sumamente especial. Pruebe cambiando el valor de A
10 A=11:GRAPHIC1,1:FORJ= 0TO360STEPA:BOX1,0,0,319, 199,J,0:NEXT:REM CAMBIE A**

Detectando teclas en la C-16, 64 y 128

**En estos equipos existe una dirección de memoria que puede ser consultada a través de la sentencia PEEK, la cual indica el código de la última tecla presionada.
La representaremos por LSTX, y varía de acuerdo al computador.
Para la C-64 ésta es la dirección decimal 197.
Para la C-16 es la 2038.
Para la C-128 es la 213.
De acuerdo al equipo que tengan realicen el siguiente ejemplo:
10 PRINTPEEK(LSTX): GOTO10
Donde LSTX tiene el valor correspondiente al equipo en cuestión.**

Run automático en la C-64

**POKE816,32
Podrán ejecutar sus programas directamente, con sólo presionar la tecla SHIFT y RUN simultáneamente.
Para volverlo a la normalidad, tipeen:
POKE816,165
u opriman la tecla de RESTORE y STOP simultáneamente.**

List y C-64

**Las siguientes líneas les permitirán listar el programa sin necesidad de interrumpir la ejecución del mismo.
Cambien los valores de X de acuerdo a las líneas a listar
100 POKE768,174:POKE769, 167:LISTXXX-XXXX:POKE 768,139:POKE769,227**

SALTOS CONDICIONALES E INCONDICIONALES

Continuando con esta serie de notas referentes al código máquina y Assembler, explicaremos cómo se realizan los saltos condicionales e incondicionales.

Cuando tuvimos que describir cómo se transfiere el control durante la ejecución de un programa escrito en código máquina, vimos la necesidad de calcular un valor (denominado offset), necesaria para desarrollar dicho salto.

```
.,  F01E  F0 F4      BEQ $F014
.,  F020  8C 9E 02   STY $029E
.,  F023  88         DEY
.,  F024  A5 9E      LDA $9E
.,  F026  91 F9      STA ($F9),Y
.,  F028  AD A1 02   LDA $02A1
.,  F02B  4A         LSR
.,  F02C  B0 1E      BCS $F04C
.,  F02E  A9 10      LDA #$10
.,  F030  8D 0E DD   STA $DD0E
.,  F033  AD 99 02   LDA $0299
.,  F036  8D 04 DD   STA $DD04
.,  F039  AD 9A 02   LDA $029A
.,  F03C  8D 05 DD   STA $DD05
.,  F03F  A9 81      LDA #$81
.,  F041  20 3B EF   JSR $EF3B
.,  F044  20 06 EF   JSR $EF06
.,  F047  A9 11      LDA #$11
.,  F049  8D 0E DD   STA $DD0E
.,  F04C  60         RTS
.,  F04D  85 99      STA $99
.,  F04F  AD 94 02   LDA $0294
.,  F052  4A         LSR
.,  F053  90 28      BCC $F07D
.,  F055  29 08      AND #$08
```

Debido al diseño del microprocesador del Drean Commodore (6510) sólo podemos efectuar saltos a la posición 128 delante del contador del programa y a la posición 127 "detrás" de él.

En Assembler existen dos maneras de indicar la dirección de salto. La primera utiliza, directamente, la dirección del salto. La segunda se hace a través de "label" o etiqueta. Por ejemplo, si disponemos de un programa monitor como MON64, HESMON, etc., la dirección se especifica en forma explícita. Vale decir que junto con la condición de salto (las cuales explicamos en números anteriores) se pone la dirección final.

Listado 1

```
C000 INX
C001 BNE $C008
C003 JMP $EA31
C006 NOP
C007 NOP
C008 RTS
```

Un ejemplo de ello se representa en el listado 1. Básicamente el programa incrementa el contenido del registro X. Si éste no ha llegado a cero se salta a través del BNE (Branch Not Equal) a la dirección $C008. Si, en cambio, el contenido de éste es cero, se procede a saltar en forma incondicional a la dirección $EA31.

Si este pequeño programa lo ingresamos usando alguno de los monitores antes descripto, veremos cómo luego de ingresar el mnemotécnico se codifica a su equivalente en código máquina.
Lo mismo sucede para los saltos condicionales. Si la dirección de salto supera el offset máximo, se imprimirá un signo de pregunta indicando la imposibilidad de efectuar tal traducción.
El mismo ejemplo pero ahora utilizando un editor de Assembler (para luego poder compilarlo) se representa a través del listado 2.
El editor no necesita direcciones de memoria, él solo utiliza números de línea para referenciar cada mnemotécnico dentro del programa fuente (no complilado).
Otro detalle importante es que la dirección $EA31 ha sido remplazada por la variable IRQ.
Esta es otra de las ventajas de trabajar con editores: poder definir variables y constantes.
La etiqueta (label) que utilizamos es SALTO. Así referenciamos la posición relativa para la transferencia del control del programa.
Luego, al compilar el programa, se traducen todas las variables, constantes y etiquetas a direcciones explícitas.
Siguiendo con los ejemplos vamos a utilizar uno de los más famosos: retardos de tiempo.

En Basic generalmente esto se hace a través de sentencias del tipo:
FOR D = 1 TO 1000: NEXT D
En Assembler se hace como lo indica el listado 3, el cual es una primera versión. Primero cargamos el registro X y el registro Y en forma inmediata con $FF (256).
Luego disminuimos en uno el contenido del Y. Si no es cero volvemos a disminuir el contenido.
Si es cero realizamos la misma tarea pero para el registro X.
Así se obtiene un retardo que equivale a 256+256.
El listado 4 es otra versión (tal vez la más utilizada) del mismo problema. La primera parte es igual que la anterior, cargamos X e Y con $FF.
Disminuimos el contenido del Y en uno. Si no es cero saltamos a disminuir nuevamente. Si llegamos a cero, disminuimos el registro X en uno.
Si llegó a cero finalizamos la tarea. En cambio, si no llegó a cero saltamos a cargar el registro Y nuevamente con $FF y a repetir todo su ciclo de decrecimientos.
Así obtenemos un retardo equivalente a 256 × 256.
Hasta aquí hemos visto cómo se opera con saltos condicionales.
Para trabajar con los saltos incondicionales se sigue poniendo la dirección de salto (o etiqueta).
Por ejemplo JMP RUTINA o JMP $EA31 es un ejemplo de ello. Lo mismo ocurre con los saltos indirectos y con los saltos a subrutinas. El primero se salta al contenido de la dirección especificada por el operando. Por ejemplo JMP ($0308) saltará a la dirección indicada por los contenidos de las direcciones $0308 y $0309.
En general en todos los saltos indirectos del tipo JMP (DIR), el control se transferirá al contenido de la dirección DIR (parte baja) y al contenido de la dirección DIR+1 (parte alta).
Por ejemplo si el contenido de la dirección $0308 es $60 y el contenido de la dirección $0309 es $2A y si hacemos JMP ($0308), es equivalente a hacer JMP $2A60.
La ventaja de trabajar así es que si queremos cambiar la dirección de salto sólo debemos modificar el contenido de las direcciones $0308 y 09.
El sistema operativo de los equipos Drean Commodore utiliza este método.
Con respecto a las subrutinas la forma de utilizarlas es a través de la instrucción JSR seguida por la dirección de salto (o etiqueta).
Al igual que el Basic el programa retornará a la instrucción siguiente al JSR.

Listado 2

```
100:       INX
200:       BNE SALTO
300:       JMP IRQ
400:       NOP
500:       NOP
600: SALTO RTS
700:.END
```

Listado 3

```
C000 LDX #$FF
C002 LDY #$FF
C004 DEY
C005 BNE $C004
C007 DEX
C008 BNE $C007
C00A RTS
```

Listado 4

```
C000 LDX #$FF
C002 LDY #$FF
C004 DEY
C005 BNE $C004
C007 DEX
C008 BNE $C002
C00A RTS
```

A: SUPERIOR. D: REGULAR.
B: MUY BUENO. E: MALO.
C: BUENO. F: INACEPTABLE.

PRINT SHOP

Broderbund Software PRESENTS...
THE PRINT SHOP™
BY DAVID BALSAM & MARTIN KAHN
COMMODORE 64 VERSION BY MARTIN KAHN & COREY KOSAK
COPYRIGHT 1984 PIXELLITE SOFTWARE ALL RIGHTS RESERVED

Rating total: A
Creatividad: A
Documentación: B
Valor en relación al precio: Se justifica
Mantiene el interés: Sí
Computadora: DC64
Editor: Pixellite Software

PRINT SHOP es, a diferencia de los anteriores, un programa utilitario que nos permite hacer todo tipo de tarjetas, carátulas, banderas, etc.

Comienza con una muy buena presentación, mostrando el nombre del programa juntamente con sus editores. Luego que el programa haya ingresado totalmente en memoria, se imprime el menú principal. Aquí aparecen los seis grandes títulos u opciones que el PRINT SHOP permite.

Estos son Greeting Card (confección de tarjetas), Sign (especias de carátulas), Banner (banderas, como los mensejes que muestran los aviones), Screen Magic (pantallas mágicas) y Graphic Editor (editor gráfico).

El ítem se selecciona oprimiendo las teclas que mueven el cursor. Así el ítem seleccionado se imprimirá en video inverso.

Al lado de ellos hay un cuadradito donde se representa, a través de un pequeño gráfico, la función del ítem elegido.

Otra de las cosas interesantes que presenta el PRINT SHOP es que prácticamente no se necesita manual ya que todos los comandos se imprimen en pantalla. Es sumamente fácil de manejar.

Vayamos a la primera opción, Greeting Card. A través de ella podemos confeccionar tarjetas como las de navidad, cumpleaños; etc.

Al ingresar en esta opción se nos pregunta si queremos diseñar nuestra propia tarjeta o utilizar las que se encuentran definidas.

Seleccionada la opción, siempre usando las teclas de movimiento del cursor, pasamos al siguiente punto.

En caso de decidir por la primera (diseño de nuestras propias tarjetas) tenemos que decirle al programa los detalles que tendrá la tapa de la misma.

Estos se refieren al recuadro (una línea, dos líneas, corazones, flores, estrellas, línea llena, sin bordes).

A continuación seleccionamos el gráfico o dibujito que tendrá la tapa. Aquí se pueden elegir entre sesenta gráficos distintos como ser tortas de cumpleaños, relojes, computadoras, robots, entre otros.

Continuamos seleccionando el tamaño del gráfico elegido. Este puede ser pequeño, mediano o grande. Para los dos primeros podemos decidir en qué posición de la tarjeta se imprimirá.

Siempre con el diseño de la tapa de la tarjeta, se continúa con la selección del tipo de letra que utilizaremos para escribir nuestro mensaje.

Para ello el PRINT SHOP dispone de ocho tipos de letras. Una vez seleccionada aparece un recuadro en donde tipeamos el mensaje o leyenda deseada.

El texto se va centrando automáticamente y podemos optar por hacer caracteres en tres dimensiones, líneas llenas o líneas vacías.

Culminada esta tarea debemos repetir todo el proceso pero para la parte de adentro. El PRINT SHOP dispone de la tecla "←" la que nos permite retroceder al menú anterior del actual. Así podemos redefinir valores incorrectos.

Finalmente indicamos el número de copias a realizar e imprimimos la tarjeta. Esta ocupa toda una hoja para impresora y, doblándola correctamente, tenemos la tarjeta lista para presentar.

La segunda opción del menú principal es Sign. A través de éste podemos crear especies de carátulas.

Una vez dentro de esta opción tenemos que seleccionar el borde (como hicimos antes), el gráfico, el tamaño del mismo y el tipo de letra.

La tercera opción es Letter Head. Esta nos permite definir el o los logotipos que se imprimirán como encabezamiento y pie de página. Así podremos tener nuestros propios formularios.

Continuamos con Banner. Para que entiendan lo que es esto, recuerden las publicidades que aparecen en la costa

utilizando avionetas. Esos carteles largos enganchados en la cola del avión. A través de Banners podrán realizar los mismos carteles.

Las opciones que siguen son Screen magic y Graphic Editor, que nos permiten generar extraños gráficos y diseñar los propios.

Algo que omitimos mencionar es que además de disponer de 60 diseños distintos es posible tomarlos desde otro disco.

De esta manera las posibilidades de PRINT SHOP aumentan considerablemente.

El único "pero", justificable, es que para usar este utilitario debemos tener una impresora.

De todas maneras PRINT SHOP es más que un programa utilitario, ¡es una imprenta!

COMMANDO

Rating total: A
Creatividad: A
Documentación: B
Profundidad del juego: A
Valor en relación al precio: Se justifica
Mantiene el interés: Sí
Computadora: DC 64
Editor: Capcon



Presentar a COMMANDO creemos que sería redundante, debido a la popularidad de este excelente juego.

El objetivo del mismo es aniquilar a todos los soldados enemigos que se nos crucen por el camino. Para ello disponemos de cinco hombres (especialistas) que armados con fusil y granadas deben matar al que se le cruce por delante.

El total de granadas es de cinco, las cuales se accionan con sólo oprimir la barra espaciadora. El mando de nuestro soldado es exclusivamente a través del joystick.

El juego comienza al oprimir el botón de disparo. Primero debemos romper la barrera de la primera etapa. Para ello debemos atravesar un túnel desde donde se nos tiran granadas y proyectiles desde un mortero.

De paso podemos liberar a un compañero que ha sido tomado prisionero. Por liberarlo se nos dan 1000 puntos.

Si logramos continuar tenemos que enfrentar la primera guarnición, en donde se abren las puertas y comienza a salir un número importante de soldados enemigos.

Antes de continuar tenemos que resaltar la excelente música que acompaña al desarrollo del juego. Parece como si dentro de la Drean Commodore 64 estuviera una banda de rock.

Es sumamente variada y, al finalizar el juego, sale una melodía que nos hace recordar el final de las películas cuando el héroe y la heroína, luego de pasar por los más diversos peligros, se besan bajo la luna.

Si podemos pasar la primera barrera, aparecerá un mensaje potente informándonos de tal hecho y que ahora debemos destruir la segunda barricada.

Así ingresamos en el segundo escenario de combate. Está formado por trincheras desde donde se nos dispara, aviones enemigos y blindados semi pesados.

Continuamos avanzando tratando de esquivar a dos lanza bazucas que custodian el puente que tenemos que atravesar.

En esta pantalla aparecen nidos de ametralladoras y morteros.

Si lo logramos, disparando el fusil y tirando granadas, encaramos los últimos metros hasta llegar a la segunda puerta. Sólo los valientes podrán continuar hasta la tercera puerta.

En el tercer escenario hay que atravesar la pista de aterrizaje enemiga, esquivando los lanza bazuca y minas terrestres.

El tercer fuerte, y creemos que el último, es sumamente difícil de sortear, casi imposible.

Durante el desarrollo del juego podemos reabastecernos de granadas y aumentar nuestro arsenal. A cada cierta distancia se encuentran estas preciosísimas herramientas de trabajo.

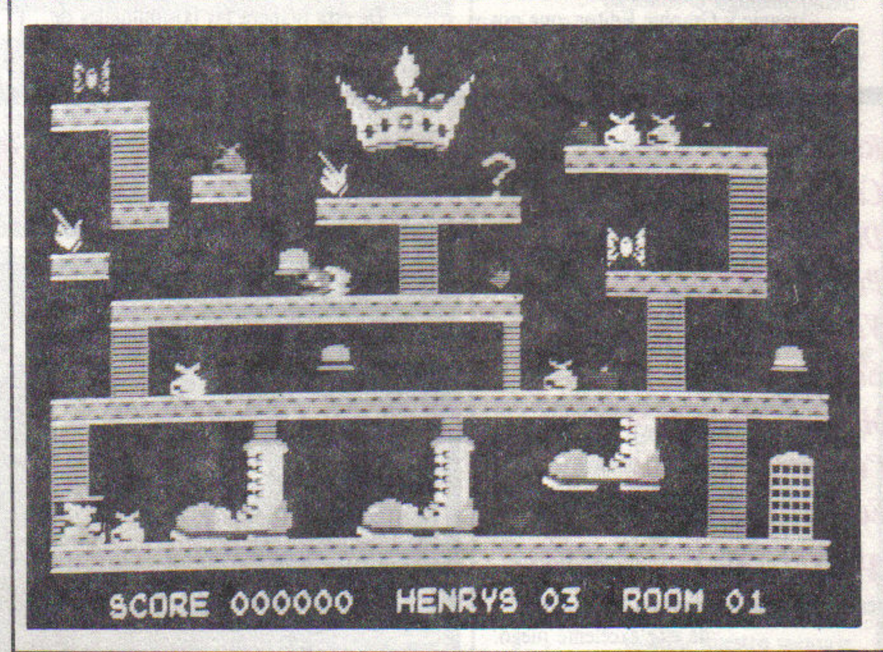
Obviamente el programa está protegido, es decir que no podemos brekearlo. Si queremos tratar de hacerlo, el programa se raya. En la pantalla aparecen una serie de rayas y la música se acelera a tal punto que se convierte en ruido. El efecto que se produce es como si se trabase una película en el proyector.

Así que no se les ocurra oprimir la tecla RESTORE junto con STOP.

COMMANDO, junto con su música, nos confunde; ¡ya que no sabemos si jugarlo o ponernos a bailar!

HENRY'S HOUSE

Rating total: B
Creatividad: B
Documentación: B
Profundidad del juego: A
Valor en relación al precio: Se justifica
Mantiene el interés: Sí
Computadora: DC 64
Editor: Chris Murray



En HENRY'S HOUSE debemos ir atravesando los ocho cuartos de la casa del actor protagónico del juego: Henry.

El primero de ellos es un hall de entrada un tanto especial. Básicamente está formado por tres zapatos gigantes que suben y bajan sincrónicamente aplastando todo lo que esté debajo de ellos, inclusive a Henry.

El objetivo del juego es el mismo en cada uno de los cuartos; es decir debemos tomar la llave que nos permite pasar a la habitación siguiente.

Esta aparecerá en pantalla solamente cuando hayamos tomado una serie de objetos "pasivos". En cada habitación hay elementos que sí pueden ser tomados y que incrementan el score y elementos que no pueden ser "comidos" y los cuales quitan una de las tres vida de Don Henry.

Como dijimos antes el primer cuarto está formado por Superzapatos que pisan todo lo que esté debajo de ellos y una especie de máscara elástica que se estira de un lado hacia otro. No hace falta decir lo que sucede si al estirarse se encuentra con Henry.

Luego de haber pasado las mayores dificultades aparecerá la llave. Debemos tomarla y dirigirnos hacia la puerta, la cual se encuentra al final del túnel formado por los "zapallos".

En caso de acceder a ella con éxito, pasaremos al segundo cuarto. Caso contrario comenzaremos el recorrido nuevamente.

La segunda habitación es el baño. Aquí se encuentran jabones, patitos (como los de chopa, chopa, chopa), esponjas, tijeras y una bañadera llena de agua donde se encuentra la llave.

Como Henry no sabe nadar, primero debemos desagotar la bañadera sacando el tapón. Hecho esto veremos cómo el agua va saliendo por las cloacas. Lamentablemente el cuerito de la canilla está roto, es decir que goteará durante el transcurso del juego (sólo en este cuarto). La llave está debajo de la canilla, por lo que el acceso se hace un tanto peligroso debido a que una gota de agua puede llegar a destruirnos.

Luego de tomar cierta cantidad de elementos, para lo cual tuvimos que saltar, subir y bajar escalares varias veces, aparecerá en acción un cepillo de dientes para tratar de "limpiar" a Henry.

Seguimos avanzando hasta llegar a la puerta que nos comunica con la siguiente habitación.

En caso de que alguien nos mate comenzaremos de nuevo pero a partir de la habitación en donde nos encontrábamos.

El tercer cuarto es la cocina de Henry. Aquí hay una tetera, tostadora, abrelatas, helados, frutillas y, cosa de locos, huevos fritos que caen desde alto y dentro de la licuadora.

Los elementos que pueden matarnos son: un impacto directo de un huevo frito, una gota (o varias gotas) de té, la caída sobre Henry de una lata de tomates desde el abrelatas y algún que otro pan tostado que nos lleve por delante.

Si podemos esquivar todo esto pasaremos al cuarto siguiente, que es la sala de estar.

Podremos ver un televisor donde se están trasmitiendo las últimas noticias aunque a veces, por problemas de antena, se pierde la imagen.

En el centro de la habitación hay un gran hogar con una inmensa chimenea.

También está la jaula de "turi", el pájaro de Henry.

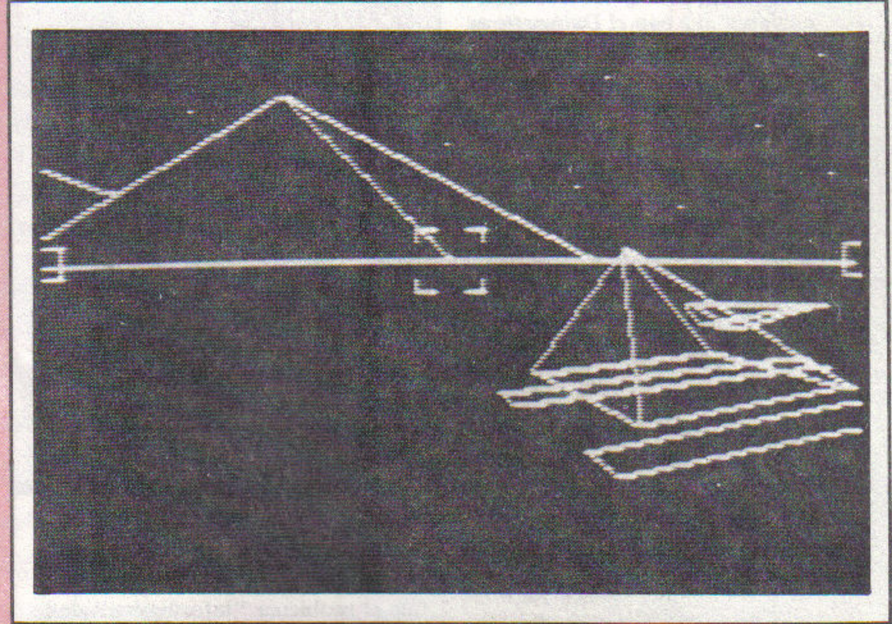
Además hay una radio, sintonizada en FM Láser, junto con muchos jarrones y velas.

En el recorrido, en busca de la llave, debemos esquivar el reloj cu-cu, la chimenea y a "turi" el cual se suelta luego de tomar una serie de jarrones y velas.

Así, el juego continúa, atravesando la pieza de los chicos (llena de juguetes), el dormitorio y el comedor, hasta que finalmente Henry llegue al sótano en donde lo esperan fantasmas, vampiros y águilas sumamente hambrientas.

Por problemas de espacio no podemos hacer una descripción detallada de HENRY'S HOUSE. Simplemente les decimos que vale la pena jugarlo.

ESTELAR 7



Rating total: B
Creatividad: A
Documentación: B
Profundidad del juego: A
Valor en relación al precio: Se justifica
Mantiene el interés: Sí
Computadora: DC 64
Editor: Software Entertainment Company

Tal vez no sea tan conocido como otros, pero consideramos que está al nivel de los mejores.

ESTELAR 7 es un juego cuyo objetivo es comandar un tanque intergaláctico para destruir las fuerzas enemigas del comandante Arcturan.

Los gráficos que se encuentran están desarrollados simulando efectos de tres dimensiones.

Al principio aparece un mensaje en donde se nos dicen las características del juego (gráficos 3D), la velocidad de los mismos durante el desarrollo del juego y los planes en desarrollar uno nuevo.

Luego de oprimir la barra espaciadora comienza a cargarse el programa principal. Aparece el menú de combate el cual nos permite seleccionar los mandos del tanque (teclado o joystick), ver los records (grandes scores) y por último, visualizar la fuerza enemiga. En esta opción se describen detalladamente cada uno de los dispositivos de ataque del comandante Arcturan.

Antes de comenzar a describirlos aparece un cartel que nos indica que la información que a continuación veremos es TOP SECRET.

El primero en aparecer es el Sandsled, el cual viene desde el fondo de la pantalla y rotando. Así crea un efecto tridimensional muy bueno.

La información que se nos da es el tipo de propulsión, el blindaje, el tipo de armas y en qué sistema se puede hallar.

El Sandsled dispone de propulsión nuclear, tiene un blindaje de 0.1 m, desarrolla una velocidad de 200 km/h y su armamento es láser de baja potencia.

El que sigue es un Láser Tank con un blindaje de 0.2 m, una velocidad máxima de 128 km/h, tracción nuclear y un armamento láser de alta potencia.

Le sigue el Hovercraff cuya propulsión es a través de un colchón de aire, 0.1 m de blindaje, armado con un cañón de luz y cuya velocidad de operación es de 200 km/h.

El cuarto es el Promler con 0.2 m de blindaje, propulsión nuclear, cañón de luz y 200 km/h de velocidad máxima.

El Heavy Tank (no hace falta traducirlo) tiene 0.3 m de blindaje, propulsado a través de un reactor nuclear. Su armamento es un cañón de luz mediano. Desarrolla una velocidad de 120 km/h.

Le sigue el Stalker del que Inteligencia no dispone de datos.

Siguen Láser Battery, Pulsar, Skimmer, Seeker, y todos diponen de gran armamento y un duro blindaje.

Como antes mencionamos debemos ir atravesando los distintos sistemas interplanetarios. Estos son System Sol, Antares, Rigel, Deneb, Sirius y Regulus.

El desafío del juego es sumamente difícil.

El comandante del tanque dispone de comandos para ampliar la situación en el escenario de combate, un mapa con la posición del enemigo y de los distintos obstáculos (cubos que no se mueven pero que molestan), y los indicadores de combustible, impactos recibidos, score parcial y del sistema.

El juego puede ser interrumpido en cualquier momento con solo oprimir la tecla de Commodore.

El movimiento del tanque es bastante rápido. Cuando avanzamos o retrocedemos, los obstáculos y/o los tanques enemigos comienzan a achicarse en función del movimiento.

Luego de haber destruido una serie de tanques enemigos aparece el Warplink, nave amiga que nos transporta de un sistema a otro.

Su forma es como una especie de cucurucho. Para ingresar dentro de él basta con chocarlo. Se escuchará una señal que indicará la colisión y comenzaremos a elevarnos.

Al mismo tiempo comenzamos a rotar velozmente y, luego de tomar una cierta altitud, se baja la compuerta de la nave transportadora.

De esta manera pasamos al siguiente sistema. El juego transcurre hasta que nuestro indicador de nivel de impactos recibidos llegue a cero.

Cada vez que esto ocurra se imprimirá el score obtenido y, en caso de superar los records, nuestro nombre será puesto en disco.

A partir del sistema Rigel aparecen dispositivos amigos para que podamos reabastecernos de combustible y de energía.

Si quieren experimentar guerra táctica y sumamente dura, ESTELAR 7 les ayudará.

CORREO-CONSULTAS

CARACTERES

En primer lugar quisiera decirles que su revista me parece extraordinaria. Comencé a adquirirla a partir del mes de abril; quisiera que me expliquen brevemente algo sobre el Assembler, también sobre los caracteres puestos entre signos de admiración.
Por último quisiera saber qué ventajas posee la C-128.
Los felicito por los programas y espero que los sigan imprimiendo en letras grandes.
Saludos a todos.

Cristian J. Martínez
Berazategui

El Assembler en un lenguaje que nos permite escribir programas en código de máquina, lo único que comprende la computadora.
Desde el número uno de nuestra publicación hemos comenzado una serie explicando cómo programar en Assembler.
Si quieres saber algo más te sugerimos que leas las notas correspondientes.
Con respecto a las ventajas de la C-128, te decimos que son muchas. La primera de ellas es la capacidad de memoria (más o menos son unos 111 kb).
Además de la cantidad de nuevas sentencias orientadas al manejo de gráficos y sonido.

TURBO

Me dirijo a ustedes con el propósito de que publiquen un programa equivalente al Turbo Tape para gestión de archivos, debido a que con el Datassette es muy engorroso para archivos largos.

Roberto Minella
Córdoba

En próximos números imprimiremos el listado del Turbo Tape y del Turbo Disk con los cuales aumentaran la velocidad del Datassette y del Drive respectivamente.

SOFTWARE PARA LA C-16

Les escribo con el fin de que me contesten cuándo van a aparecer en vuestra revista programas para la Dream Commodore 16. Desde ya les agradezco su contestación y los felicito por la revista.

Diego Batwini
Bs. As.

Poco a poco nuestra publicación irá incrementando el material (notas técnicas más programas) para la Drean Commodore 16.
Por ahora les pedimos a los usuarios de dicha computadora un poco más de paciencia.
De todas maneras queremos decirles que el problema "informativo" de la Drean Commodore 16 es mundial.
Igualmente nosotros estamos haciendo todo lo que está a nuestro alcance para investigar sobre esta computadora y, así, poder brindarles los frutos de ellas en notas y programas.

Continuamos con esta sección para que los lectores planteen sus consultas y sugerencias. Para eso deben escribir a Revista para usuarios de Drean Commodore, Paraná 720, 5to. Piso, (1017) Cap. Federal.

UNA ACLARACION IMPORTANTE

En el número anterior hemos realizado la revisión del programa TRUCO sin especificar quién o quiénes fueron los "sabios" creadores de este excelente juego.

Al no poder encontrar en el programa los datos del autor, ya que los *"piratas"* se encargaron de borrarlos, tuvimos que poner un "?" en respuesta al editor del mismo.

Hecha la aclaración vayamos a lo concreto: El editor del programa *TRUCO* es *Electrónica Leo* y su autor es *Miguel Grimberg* un joven de 17 años, a quien felicitamos.

JULIO 1986 Nº 16 A 2,30 REP. ARGENTINA

K64

COMPUTACION PARA TODOS

Desarrollos:

Grabador de Eproms

Sistema de Luces

Suplemento

Educativo para CZ, TK, C64, TI y MSX

MSX: Software Comercial

15 Programas Inéditos

Concurso: Ultimo Mes

Archivo para la Commodore 16